Oliver Kloos

Generierung von Simulationsmodellen auf der Grundlage von Prozessmodellen

Generierung von Simulationsmodellen auf der Grundlage von Prozessmodellen

Oliver Kloos

Universitätsverlag Ilmenau
2014

Impressum

Bibliografische Information der Deutschen Nationalbibliothek
Die Deutsche Nationalbibliothek verzeichnet diese Publikation in der Deutschen Nationalbibliografie; detaillierte bibliografische Angaben sind im Internet über http://dnb.d-nb.de abrufbar.

Diese Arbeit hat der Fakultät für Wirtschaftswissenschaften der Technischen Universität Ilmenau als Dissertation vorgelegen.

Tag der Einreichung:	8. Mai 2012
1. Gutachter:	Univ.-Prof. Dr. Volker Nissen (Technische Universität Ilmenau)
2. Gutachter:	Univ.-Prof. Dr. Steffen Straßburger (Technische Universität Ilmenau)
Tag der Verteidigung:	25. Februar 2013

Technische Universität Ilmenau/Universitätsbibliothek
Universitätsverlag Ilmenau
Postfach 10 05 65
98684 Ilmenau
www.tu-ilmenau.de/universitaetsverlag

Herstellung und Auslieferung
Verlagshaus Monsenstein und Vannerdat OHG
Am Hawerkamp 31
48155 Münster
www.mv-verlag.de

ISBN 978-3-86360-086-0 (Druckausgabe)
URN urn:nbn:de:gbv:ilm1-2013000606

Titelfoto: photocase.com | Nortys

Inhaltsverzeichnis

Abbildungsverzeichnis

Tabellenverzeichnis

1 Einleitung

1.1 Problemstellung und Zielsetzung der Arbeit

Bei jedem Simulationsvorhaben stellt sich die Aufgabe, ein passendes Simulationsmodell als Grundlage zu erzeugen. Wenn Geschäftsprozesse Gegenstand des Simulationsvorhabens sind, bilden diese die Grundlage für das Simulationsmodell (Desel und Erwin 2000, S. 129; An und Jeng 2005, S. 2068). In Unternehmen können diese Geschäftsprozesse als modellierte Modelle – als Geschäftsprozessmodelle – vorliegen. Dies wird als zentrale Annahme der Arbeit unterstellt. Werden modellierte Geschäftsprozessmodelle in Simulationsmodelle überführt, kristallisieren sich drei Problemfelder heraus: Die Reduktions-, Erweiterungs- und Transformationsproblematik.

Wie aus dem Modellbegriff[1] hervorgeht, sind drei Merkmale zu berücksichtigten, das Abbildungs-, das Verkürzungs- und das pragmatische Merkmal. Die Abbildung des Geschäftsprozessmodells und daraus resultierende Simulationsmodells basiert auf dem gleichen Gegenstand, dem Geschäftsprozess. Unterschiede hinsichtlich des Verkürzungsmerkmals ergeben sich aus dem pragmatischen Merkmal. Ein Geschäftsprozessmodell wird beispielsweise erstellt, um Schwachstellen in den Unternehmensabläufen aufzudecken, für die Dokumentation des geschäftlichen Ablaufs oder die Anforderungsanalyse und Umsetzung eines Informationssystems (Gadatsch 2010, S. 112). Von den genannten Beispielen wird lediglich das Aufdecken der Schwachstellen ebenfalls von Simulationsmodellen abgedeckt; beide Modelle beziehen sich aber auf unterschiedliche Schwachstellen. Während mit Simulationsmodellen Auslastungen und Durchlaufgeschwindigkeiten, und damit verbundene Schwachstellen und Engpässe, ermittelt werden (Allweyer 2005, S. 257), werden bei Geschäftsmodellen Organisationsbrüche oder Datenredundanzen

1 Der Modellbegriff wird auf Seite 17 erläutert.

(Allweyer 2005, S. 228) identifiziert. Aus der Verwendung der Geschäftsprozessmodelle leitet sich ab, welche Informationen aus Sicht des Verkürzungsmerkmals bei einem Geschäftsprozessmodell benötigt werden. Zum einen werden Daten und Systeme berücksichtigt, zum anderen wird eine Stelle bei den auszuführenden Tätigkeiten angegeben. Ebenfalls werden aufgrund des Gedankens der Dokumentation Tätigkeiten, die von der gleichen Stelle ausgeführt werden, teilweise sehr detailliert beschrieben. Das Simulationsmodell hingegen dient der Analyse der Ausführung und nicht der Struktur. Welche Ressourcen in welchen Mengen und zu welcher Zeit benötigt werden stellen relevante Informationen dar. Schwerpunkt eines Simulationsmodells liegt daher bei den Ressourcen, unabhängig ob diese menschlich, nicht menschlich, stationär oder beweglich sind. Werden gleiche Ressourcen benötigt, so können Arbeitsschritte im Simulationsmodell zusammengefasst werden.

Aus Sicht der drei Merkmale des Modellbegriffs ergeben sich die ersten zwei Problemfelder. Geschäftsprozessmodelle weisen einerseits einen höheren Detaillierungsgrad hinsichtlich der Beschreibung der Prozessschritte auf, andererseits benötigten Simulationsmodelle quantifizierte sowie detailliertere Daten (Dickmann et al. 2007, S. 277; Kloos et al. 2009, S. 84-85 und Kloos et al. 2010, S. 92). Werden Geschäftsprozessmodelle als Ausgangspunkt für Simulationsmodelle verwendet, müssen diese zum einen reduziert und zum anderen erweitert werden.

Wenn das Geschäftsprozessmodell und das Simulationsmodell den gleichen Gegenstand beschreiben, dann müssten beide Modelle in das jeweils andere Modell überführt werden können. Aus pragmatischer Sicht müssen zur vollständigen Beschreibung hierzu noch Anpassungen – die Reduktion und Erweiterung – vorgenommen werden. Wenn eine Überführung möglich wäre, dann muss diese Überführung durch entsprechende Regeln auch automatisiert werden können, vorausgesetzt beide Modelle liegen in einer auslesbaren Form vor. Hier entsteht das dritte Problemfeld der Überführung eines Geschäftsprozessmodells in

ein Simulationsmodell: Geschäftsprozessmodelle können mit unterschiedlichen Notationen modelliert werden; Form und Funktionalität eines Simulationsmodells werden von den Gestaltungsmöglichkeiten einer Simulationsumgebung bestimmt.

Bei einer direkten Überführung von einem Prozessmodell in ein Simulationsmodell entstehen eine Vielzahl an Transformationstheorien. Die Anzahl der notwendigen Transformationstheorien[2] wäre $n \cdot m$. Alternativ zu einer direkten Überführung kann eine indirekte Überführung über eine zentrale Instanz vollzogen werden (Kloos und Nissen 2010, S. 111). Geschäftsprozessmodelle werden in die zentrale Instanz überführt und diese wird in Simulationsmodelle überführt. Hierdurch beträgt die Anzahl der notwendigen Transformationstheorien $n+m$. Sobald $\{(n,m) \in \mathbb{N} \mid (n>2) \vee (m>2)\}$ wahr ist, ist die indirekte Überführung hinsichtlich der Transformationstheorien mit weniger Aufwand verbunden, als die direkte Überführung. Eine zentrale Instanz bietet zusätzlich die Möglichkeit, für die Ausführung relevante aber nicht modellierte Elemente hinzuzufügen, die nicht adäquat modelliert wurden (Kloos et al. 2009, S. 92-94).

Um Informationsverluste bei der Überführung zu vermeiden, muss die zentrale Instanz eine semantische Obermenge der berücksichtigen Quell- und Zielsprachen bilden, wie Decker et al. (2009, S. 93)[3] mit Bezug auf Vanderhaeghen et al. (2005) schlussfolgern. Decker et al. (2009) unterstellen bei Ihrem Argument aber den gleichen Verwendungszweck des Quell- und Zielmodells. Im Hinblick auf unterschiedliche Verwendungszwecke wird die These vertreten: Ein Informationsverlust bei der Überführung ist erlaubt; es müssen nur die Informationen überführt werden, die im Zielmodell – für die Simulation eines Geschäftsprozesses – relevant sind. Als Konsequenz

2 Die Zahl n steht für die Anzahl der Geschäftsprozessnotationen und die Zahl m für die Anzahl der Simulationsumgebungen.

3 Decker et al. (2009) sprechen nicht von einer zentralen Instanz, sondern von einer dritten Sprache, die verwendet wird, um ein Quellmodell in ein Zielmodell zu überführen. Im Fall der Überführung einer EPK in BPMN seien daher Petrinetze und BPEL nicht geeignet.

dieser These ist eine Rücktransformation in das Ursprungsmodell nicht möglich. Entsprechend ist Ziel der Arbeit nur eine einseitige Überführung, mit Geschäftsprozessmodellen als Ausgangspunkt und Simulationsmodellen als Zielpunkt.

Aus diesen Ausführungen leitet sich die zentrale Fragestellung dieser Arbeit ab:

Wie können Geschäftsprozessmodelle mittels einer zentralen Instanz in Simulationsmodelle überführt werden?

Abgeleitet aus der These von Decker et al. (2009, S. 93) muss – aus Sicht des Abbildungsmerkmals – eine zentrale Instanz eine semantische Obermenge der berücksichtigten Notationen für Geschäftsprozess- und Simulationsmodelle sein. Das Modell dieser zentralen Instanz wird im Kontext dieser Arbeit als Transformationsmodell bezeichnet. Aus der These leiten sich zwei Hypothesen für das Transformationsmodell ab:

H1 Wenn alle Elemente des Geschäftsprozessmodells in ein Transformationsmodell überführt werden können, dann ist das Transformationsmodell eine semantische Obermenge der betrachteten Geschäftsprozessmodellnotationen.

H2 Wenn alle Elemente des Transformationsmodells in die Notation der Simulationsmodelle überführt werden können, dann ist das Transformationsmodell eine semantische Obermenge der betrachteten Simulationsmodelle.

Beide Hypothesen H1 und H2 betrachten eine Transformation eines Quellmodells in ein Zielmodell. Eine isolierte Betrachtung eines Elements des Quellmodells unterschiedlich interpretiert werden kann (Decker et al. 2009, S. 100). Ein Element muss im semantischen Kontext des Modells betrachtet werden, um eindeutig bestimmbar zu sein. Auf dieser Annahme baut die dritte Hypothese auf:

H3 Wenn im Kontext des Transformationsmodells jedes semantische Konstrukt einer Quellsprache in ein semantisch identisches Konstrukt einer Zielsprache überführt werden kann, kann die Transformation automatisch erfolgen.

In Bezug auf das Abbildungsmerkmal fordert diese Hypothese: Ein semantisches Konstrukt in der Quellsprache muss einem semantischen Konstrukt der Zielsprache entspricht. Implizite angenommen wird ein semantisch identischer Sachverhalt aus simulationstechnischer Sicht. Damit eine automatische Transformation möglich wird, muss ein semantisches Konstrukt identifiziert werden und das Quellmodell syntaktisch korrekt sein. Wird auf ein syntaktisch korrektes Modell eine automatische Transformation angewendet, muss das Zielmodell ebenfalls ein syntaktisch korrektes Modell sein. Bezogen auf die anfangs aufgestellten Problemstellungen adressieren die ersten drei Hypothesen das dritte Problemfeld: Die Transformationsproblematik.

Aus der zentralen Fragestellung leitet sich im Kontext der ersten zwei Problemfelder – der Reduktions- und Erweiterungsproblematik – eine weitere Fragestellung ab:

Wie kann das Transformationsmodell angepasst werden,
damit es in ein Simulationsmodell überführt werden kann?

Wie in der zentralen Annahme formuliert, werden modellierte Geschäftsprozessmodelle für das Transformationsmodell vorausgesetzt. Eine Änderung eines Geschäftsprozessmodells ist nicht vorgesehen. Das Simulationsmodell hingegen soll alle simulationsrelevanten Daten enthalten. Die Reduktions- und Erweiterungsproblematik muss daher im Rahmen des Transformationsmodells adressiert werden. Methoden sind notwendig, die sowohl die Abstraktion des Modells, als auch die Erweiterung um simulationsrelevante Aspekte, wie auch die Detaillierung mit quantitativen Daten umfassen. Diese, die Elemente direkt betreffenden, Methoden können der Syntax zugeordnet werden. Elemente in Geschäftsprozessmodellen weisen aber nicht nur eine

Syntax sondern auch semantische Bezeichnungen auf. Es wird die These aufgestellt: Aus Bezeichnungen können simulationsrelevante Informationen abgeleitet werden, welche zur Lösung der Reduktions- und Erweiterungsproblematik verwendet werden können. Auf Grundlage dieser These wird die nachfolgende Hypothese aufgestellt:

H4 Wenn die Bezeichnungen der Elemente der auszuführenden Tätigkeiten Informationen über den Geschäftsprozess beinhalten, dann können diese für die Beschaffung von simulationsrelevanten Informationen verwendet werden.

Bezeichnungen können auch aus grammatikalischer Sicht betrachtet werden. Hierfür müssen einzelne Wortarten bei den Tätigkeiten analysiert werden, woraus sich die zu belegende Hypothese ergibt:

H5 Wenn die Bezeichnungen der Elemente der auszuführenden Tätigkeiten in einem Geschäftsprozessmodell hinsichtlich ihrer Wortart betrachtet werden, dann können daraus Informationen für die Simulation eines Geschäftsprozessmodells abgeleitet werden.

Im Rahmen der vorliegenden Arbeit sollen die zwei gestellten Fragestellungen beantwortet sowie die fünf aufgestellten Hypothesen überprüft werden. Daraus leiten sich die nachfolgenden fünf Zielstellungen für die Arbeit ab:

1. Ein Transformationsmodell entwickeln, um Geschäftsprozessmodelle in Simulationsmodelle zu überführen.

2. Transformationsregeln herleiten, um Geschäftsprozessmodelle automatisch in das Transformationsmodell zu überführen.

3. Regeln und Methoden aufstellen, um das Transformationsmodell mit simulationsrelevanten Daten anzureichern und auf die wesentlichen Aspekte zu reduzieren.

4. Wortarten in den Bezeichnungen des Transformationsmodells untersuchen, um daraus Erkenntnisse für die Beschaffung simulationsrelevanter Daten abzuleiten.

5. Transformationsregeln herleiten, um das Transformationsmodell automatisch in Simulationsumgebungen zu überführen.

Bevor auf die Methodik eingegangen wird, legt der nachfolgendem Abschnitt die wissenschaftliche Grundposition, an die sich die Arbeit orientiert, dar.

1.2 Wissenschaftstheoretische Grundposition

Die wissenschaftstheoretische Grundposition orientiert sich an Popper (1995) und Kuhn (1993). Die Ausarbeitungen von Popper bilden die Grundlagen für die wissenschaftliche Methodik, während Kuhn einen Ordnungsrahmen für diese bereitstellt.

Eine Kernaussage von Popper (1995, S. 9) besagt, dass „[...] alle Gesetze oder Theorien als Hypothesen oder Vermutungen [...]“ betrachtet werden müssen. Eine empirische Rechtfertigung einer Theorie ist nach Popper (1995, S. 13-17) nicht möglich; es können lediglich Gründe gesucht werden, warum eine Theorie der anderen überlegen ist. Für die Bevorzugung von Theorien führt Popper (1995, S. 18) den „Bewährungsgrad“ ein. Dieser umfasst die Punkte „wie die Theorie ihre Probleme löst; der Grad ihrer Prüfbarkeit; die Strenge der Prüfungen, der sie unterzogen wurde; und wie sie diese Prüfungen bestanden hat. Der Bewährung(sgrad) ist also ein bewerteter Bericht über die bisherigen Leistungen“. Eine Theorie kann demnach nicht bewiesen, sondern lediglich widerlegt werden (Popper, 1995, S. 30).

Das Hauptziel der Wissenschaft sollte die Suche nach Wahrheit sein (Popper 1995, S.44). Es sollen Theorien aufgestellt werden, die der Wahrheit näherkommen. Hierbei führt Popper auf, „in meinen Augen ist das Streben nach Einfachheit und Durchsichtigkeit eine moralische

Pflicht aller Intellektuellen: Mangel an Klarheit ist eine Sünde, Aufgeblasenheit ein Verbrechen“. Ein Näherkommen an die Wahrheit sollte demnach aus moralischer Sicht eine Einfachheit und Klarheit aufweisen. Hierfür wird immer das Hintergrundwissen herangezogen, dass zum gegenwärtigen Zeitpunkt als unproblematisch angenommen wird (Popper 1995, S. 70-72).

Mittels Induktion, als wissenschaftliche Methode, können Hypothesen aufgestellt, aber keine Sicherheit über die Wahrheit der Hypothese erlangt werden (Popper 1995, S. 99-101). Popper spricht indirekt ein weiteres Problem bei den Hypothesen an. Werden keine Annahmen für die Hypothesen angegeben, so sind diese überall und zu jedem Zeitpunkt gültig. Annahmen schränken den Gültigkeitsbereich einer Hypothese ein, sorgen jedoch nicht für eine höhere Sicherheit hinsichtlich der Wahrheit einer Hypothese.

Wissenschaftlicher Fortschritt zeichnet sich nach Popper (1995, S. 125) durch „die Formulierung von Problemen, das Auftauchen neuerer Problemsituationen, konkurrierende Theorien, wechselseitige Kritik durch Argumentation [...]“ aus. An diesem Punkt setzen die Ausführungen von Kuhn an. In ihren Anfängen befindet sich eine Wissenschaft in einem Zustand der Vorwissenschaft (VW). Wenn diese sich entwickelt, geht sie in eine normale Wissenschaft (NW) über. Die normale Wissenschaft weist ein vorherrschendes Paradigma auf, mit dem Fragen oder Probleme gelöst werden können. Steht die normale Wissenschaft aber Problemen gegenüber, die mit dem vorherrschenden Paradigma nicht gelöst werden können, so gerät die Wissenschaft in eine Krise (K). Aus dieser Krise führt nach Kuhn eine Revolution (R), welche ein neues Paradigma hervorbringt, wodurch sich eine neue normale Wissenschaft etabliert. Dieser wissenschaftliche Fortschritt kann mit dem nachfolgenden Schema beschrieben werden:

$$VW \rightarrow NW_1 \rightarrow K_1 \rightarrow R_1 \rightarrow NW_2 \rightarrow K_2 \rightarrow ...$$

Als eine normale Wissenschaft bezeichnet Kuhn (1993, S. 25) „eine Forschung, die fest auf einer oder mehreren wissenschaftlichen Leistungen der Vergangenheit beruht, Leistungen, die von einer bestimmten wissenschaftlichen Gemeinschaft eine Zeitlang als Grundlage für ihre weitere Arbeit anerkannt werden." Ein Paradigma einer Normalwissenschaft entsteht, wenn wissenschaftliche Leistungen eine gewisse Neuheit aufweisen, „[...] um eine beständige Gruppe von Anhängern anzuziehen, die ihre Wissenschaft bisher auf andere Art betrieben hat, und gleichzeitig war sie noch offen genug, um der neuen Gruppe von Fachleuten alle möglichen ungelösten Probleme zu stellen" Kuhn (1993, S. 25). Ein Paradigma umfasst Gesetze, Theorien, Anwendungen und Hilfsmittel. Diese werden, in der Regel, in den wissenschaftlichen Lehrbüchern einer Disziplin aufgeführt. Eine Vorwissenschaft hingegen kennzeichnet sich durch das Fehlen eines Paradigmas oder eines geeigneten Kandidaten dafür. So werden in der Vorwissenschaft beobachtete Phänomene, von den Forschern unterschiedlich beschrieben und aufgefasst (Kuhn 1993, S. 30-32).

Das Thema dieser Arbeit ist in den Bereich der Wirtschaftsinformatik einzuordnen. Nachfolgend wird aus Sicht Kuhns Paradigmata die Wirtschaftsinformatik betrachtet, um ein geeignetes Paradigma für die Arbeit zu bestimmen.

Nach Heinrich et al. (2007, S. 14) ist „eine weitgehend akzeptierte Umschreibung des Gegenstandbereichs der Wirtschaftsinformatik […] die Wissenschaft von Informations- und Kommunikationssystemen in Wirtschaft und Verwaltung". Informations- und Kommunikationssysteme werden meist nur mit der Bezeichnung Informationssystem abgekürzt, wie dies auch in dieser Arbeit erfolgt. Ein Informationssystem ist die Kombination der drei Faktoren, Informations- und Kommunikationstechnik, Mensch und Aufgabe (Heinrich et al. 2007, S. 16). Unter anderem bezeichnet Hess (2010, S. 7) ein Informationssystem als sozio-technisches System, dass sich, wie bei Heinrich et al. (2007), aus den drei Komponenten Aufgabe, Mensch als personeller Aufgabenträger und IT als maschineller Aufgabenträger bildet. Die

Informations- und Kommunikationstechnik, wie von Heinrich et al. (2007) beschreiben ist daher ebenfalls ein Aufgabenträger.

Schwerpunkt der Arbeit ist die Generierung von Simulationsmodellen auf der Grundlage von Prozessmodellen. Auch wenn der Fokus der Arbeit auf der Generierung von Simulationsmodellen liegt, wird dennoch ein Informationssystem gestaltet. In einer Analogie zwischen Informationssystem und Simulationsmodell kann ein direkter Bezug auf den Gegenstandsbereich der Wirtschaftsinformatik vollzogen werden. Die drei Komponenten eines Informationssystems nach Hess (2010, S. 7) sind Gegenstand der Analogie. Ziel einer Simulationsstudie sind Erkenntnisse über das Verhalten eines Systems, wofür ein Simulationsmodell und eine adäquate Simulationsumgebung benötigt werden. Wird dies als Informationssystem betrachtet, so ist die Aufgabe die Erkenntnis über das Systemverhalten. Das Simulationsmodell stellt die Grundlage für eine Simulationsstudie dar. Dieses muss mit entsprechenden Daten angereichert werden. Mit den Daten werden Experimente im Rahmen der Simulationsstudie durchgeführt. Einige Daten müssen für die Experimente angepasst oder abhängig von den Ergebnissen eines Experimentes geändert werden. Wenn das Ziel der Simulationsstudie die Verbesserung des betrachteten Systems ist, so ist ein Mensch notwendig, der die Ergebnisse bewertet. Der Mensch tritt an dieser Stelle als personeller Aufgabenträger auf. Die eigentliche Durchführung der Simulationsstudie erfolgt mittels der Simulationsumgebung. Hierbei handelt es sich um eine Anwendungssystem, also eine Software. Die Software kann als IT-Komponente und somit maschineller Aufgabenträger aufgefasst werden. Sie übernimmt die Berechnung der Simulationsstudie unter den angegebenen Parametern.

Die Analogie verdeutlicht die Durchführung einer Simulationsstudie mittels eines Informationssystemen. Eine ähnliche Analogie kann aber auch für die Generierung von Simulationsmodellen mittels des Transformationsmodells – auf der Grundlage von Prozessmodellen – vollzogen werden. Die Aufgabe ist die Generierung. Der menschliche Aufgabenträger reichert das Transformationsmodell mit simulations-

relevanten Daten an oder reduziert zu komplexe Abläufe. Die IT als maschineller Aufgabenträger realisiert die Transformation und unterstützt den Menschen mittels geeigneter Methoden und Verfahren. Das Thema der Arbeit kann demnach als Gegenstand der Wirtschaftsinformatik aufgefasst werden.

Aus wissenschaftlicher Sicht kann die Wirtschaftsinformatik in zwei Teildisziplinen unterteilt werden, in die gestaltungsorientierte und die verhaltensorientierte Wirtschaftsinformatik. Die gestaltungsorientierte Wirtschaftsinformatik betrachtet die Konstruktion von Informationssystemen und hat als Ziel normative und praktisch verwendbare Ziel-Mittel-Aussagen. Im Gegensatz dazu betrachtet die verhaltensorientierte Wirtschaftsinformatik das Informationssystem direkt und fokussiert auf die Entdeckung von Ursache-Wirkungs-Zusammenhängen (Österle et al. 2010a, S. 666-667). Auf der Grundlage der obigen Analogie, kann die Generierung eines Simulationsmodells als die Generierung eines Informationssystems aufgefasst werden. Das Thema der Arbeit ist somit der gestaltungsorientierten Wirtschaftsinformatik zuzuordnen.

In der deutschsprachigen Wirtschaftsinformatik wird die gestaltungsorientierte Ausrichtung gegenwärtig als konsensfähiges Forschungsparadigma diskutiert (Riege et al. 2009, S. 69). Dieser Eindruck wird gestützt durch das „Memorandum zur gestaltungsorientierten Wirtschaftsinformatik“ von Österle et al. (2010)[4]. Neben den zehn Autoren dieses Memorandums werden 111 Professoren als Mitunterzeichner aufgeführt (Österle et al. 2010a, S. 669-672), die „[...] die darin formulierten Grundsätze unterstützen“. Aussagen im Memorandum können daher für die Gestaltung von wissenschaftlichen Arbeiten in der gestaltungsorientierten Wirtschaftsinformatik herangezogen werden. Wie aber Riege et al. (2009, S. 69) formulierten, ist dies erst ein Kandidat für ein Paradigma. Entsprechend wird die These

4 Das Memorandum zur gestaltungsorientierten Wirtschaftsinformatik ist auch in Österle et al. (2010b) erschienen. Neben dem Memorandum enthält dieses Werk ergänzende Beiträge, welche Aspekte der gestaltungsorientierten Wirtschaftsinformatik vertiefen.

aufgestellt, dass sich die Wirtschaftsinformatik gegenwärtig noch im Zustand der Vorwissenschaft befindet, sich aber auf dem Weg begibt, zu einer Normalwissenschaft zu werden. Diese These wird durch die Aussage von Riege et al. (2009, S. 70) gestützt: „Darüber hinaus verstehen einige Autoren [...] auch Theorien im Sinne eines Artefakts. Hier wird dem Charakter einer reifen Wissenschaft Rechnung getragen, die es vermag, Theorien zu schaffen." Entsprechend dieser Aussage sieht sich die Wirtschaftsinformatik gerne als reife Wissenschaft, hat diesen Zustand aber noch nicht erreicht hat.

Aus Sicht einer Vorwissenschaft weist die Wirtschaftsinformatik aber bereits einen geeigneten Kandidaten für ein Paradigma vor. Das Memorandum mit den Mitunterzeichnern ist ein Beleg hierfür (Österle et al. 2010a). Aber wie bereits zu Kuhn (1993, S. 30-32) Aussagen ausgeführt, bedarf es für den Übergang von einer Vorwissenschaft zur Normalwissenschaft eine einheitliche Begriffsbasis. Bezogen auf die Teildisziplin der Wirtschaftsinformatik, die sich mit Geschäftsprozessen beschäftigt, ist festzustellen, dass bei der Betrachtung der Einführungsliteratur keine einheitliche Definition vorgefunden werden kann (beispielsweise: Gadatsch 2010, S. 40-41; Allweyer 2005, S. 51-54; Becker und Schütte 2004, S. 107). Diese Bedingung für eine Normalwissenschaft ist noch nicht gegeben.

Mit der Formulierung des Erkenntnisziels und der Ergebnistypen ist im Memorandum (Österle et al. 2010a, S. 666-667) aber der Weg bereitet, damit sich mehrere Forscher mit ungelösten Problemen beschäftigen können. Der entscheidende Punkt, der aber ein Paradigma nach Kuhn ausmacht, ist das Vorhandensein einer Theorie, auf der weitere wissenschaftliche Erkenntnisse aufbauen. Mögliche Theorien, die als Grundlage dienen können werden in der Forschungsgemeinschaft der Wirtschaftsinformatik bereits seit einigen Jahren diskutiert (beispielsweise: Lehner 1999; Greiffenberg 2003; Wilde und Hess 2006; Zelewski 2009; Houy et al. 2009; Loos et al. 2011). Mögliche Theorien als Grundlage für die gestaltungsorientierte Wirtschaftsinformatik werden im Memorandum (Österle et al. 2010a) aber nicht genannt. Stattdessen

wird ein Methodenpluralismus vertreten (Österle et al. 2010a, S. 666). Aus Sicht einer Theorie als Grundlage könnten sich mit dem Methodenpluralismus mehrere Teildisziplinen in der Wirtschaftsinformatik bilden, die jeweils ihre eigenen Theorien zu Grunde legen. Ohne diese Theorie, als Grundlage für wissenschaftliche Arbeiten im Kontext eines Paradigmas, hat die Wirtschaftsinformatik den Stand einer Normalwissenschaft aber noch nicht erreicht.

Diese Ausführungen sollen die formulierte These unterstützten, dass sich die Wirtschaftsinformatik gegenwärtig noch im Zustand der Vorwissenschaft befindet. Aufgrund der fehlenden Theorie werden der Modellbegriff und die allgemeine Systemtheorie als theoretische Basis für die Arbeit herangezogen.

1.3 Methodik und Aufbau der Arbeit

In diesem Abschnitt erfolgt eine Erläuterung des Aufbaus der Arbeit. Die verwendete Methodik – orientiert an Popper (1995) – wird in dieser integriert geschildert.

Das erste Ziel der vorliegenden Arbeit ist die Erstellung eines Transformationsmodells. Die notwendigen theoretischen Grundlagen werden in in Kapitel 2 gelegt. Hierfür werden zunächst in Kapitel 2.1, basierend auf einschlägiger Literatur, der Modellbegriff sowie systemtheoretische Grundlagen behandelt. Anschließend erfolgt in Kapitel 2.2 die Definition der zwei zentralen Begriffe der Arbeit: Prozess- und Simulationsmodell. Um den Begriff des Prozessmodells zu definieren, werden, im Sinne von Kuhn, Lehrbücher sowie oft zitierte Werke herangezogen. Für den Begriff des Simulationsmodells wird die in der Ingenieurwissenschaft am häufigsten verwendete Definition der VDI herangezogen.

Als zweites Ziel der Arbeit sollen Regeln aufgestellt werden, um Geschäftsprozessmodelle in das Transformationsmodell zu überführen. Hierfür werden in den Unterkapiteln von 2.3 drei Notationen für die Modellierung von Geschäftsprozessen aufgeführt. Alle Elemente, die an dieser Stelle aufgeführt sind, werden im Rahmen der Transformationsregeln berücksichtigt. Vorrangig werden die Spezifikationen der Notationen herangezogen, aber auch auf einschlägige Literatur zurückgegriffen. Daraufhin wird in Kapitel 2.4 das Thema der Simulierbarkeit von Prozessmodellen untersucht. Hier wird der Frage nachgegangen, wann Prozesse simulierfähig sind, aber auch wann ein Prozess simulationswürdig ist. Daran anschließend folgt eine Betrachtung der wissenschaftlichen Literatur, bei der Ansätze für die Simulation von Geschäftsprozessen untersucht werden. Die Art der Literaturrecherche wird an dieser Stelle aufgeführt. Das Kapitel endet mit einer Offenlegung der Forschungslücke. In den Unterkapiteln von 2.5 wird die Notation von zwei Simulationsumgebungen betrachtet, für die gemäß des fünften Zieles Transformationsregeln aufzustellen sind. Weitere Simulationsumgebungen, die im Rahmen der Forschungsarbeit betrachtet wurden, werden in Kapitel 3.6.3 kurz dargelegt. Es werden Probleme aufgeführt, weshalb die Simulation eines Geschäftsprozessmodells nicht erfolgen konnte und welche Aspekte weiter untersucht werden sollten, um eine Transformation zu ermöglichen.

Das nach dem ersten Ziel zu entwerfende Transformationsmodell wird in Kapitel 3 behandelt. Hierfür wird in Kapitel 3.1 das Transformationsmodell sowie die Sichten und Modelle, die dieses umfasst, hergeleitet. Das Konzept wird im anschließenden Kapitel 3.2 aus Sicht eines zentralen Ansatzes sowie der Model Driven Architecture betrachtet. Anhand dieser Analogie soll eine Einordnung des Konzeptes in artverwandte Ansätze erfolgen. In diesem Zusammenhang erfolgt die Erläuterung der Regelbasis sowie deren Klassifizierung. Im Sinne der Durchführung einer Simulationsstudie ist im darauffolgenden Kapitel 3.3 Konzept in Vorgehensmodelle zur Durchführung von Simulationsstudien eingeordnet. Die Unterkapitel zu 3.4 adressieren direkt das zweite Ziel der Arbeit. Basierend auf den Spezifikationen

und den Notationelementen, in den Unterkapiteln von 2.3, werden Transformationsregeln hergeleitet, die automatisch ein Geschäftsprozessmodell in das ProSiT Ablaufdiagramm überführen. Hierzu wird der semantische Kontext der Elemente einer Notation verwendet, um die Transformationsregeln deduktiv aus den Spezifikationen herzuleiten. Das Ziel drei und vier, die Erstellung von Methoden und der Sprachanalyse, um simulationsrelevante Informationen abzuleiten, wird in den Unterkapiteln von 3.5 in Angriff genommen. Die Methoden und Verfahren, für das dritte Ziel werden induktiv hergeleitet. Hierzu wurden konkrete Ausprägungen von Geschäftsprozessen, die in das Transformationsmodell überführt wurden untersucht, um Regeln abzuleiten, die auch bei anderen Geschäftsprozessen angewendet werden können, um das Transformationsmodell zu reduzieren oder zu erweitern. Für die Regel und Verfahren, die auf der Untersuchung der Wortarten herrühren, wurden ebenfalls konkrete Ausprägungen des Transformationsmodells verwendet. Hierbei wird von einer konkreten Ausprägung eines Wortes abstrahiert und statt dessen die Wortart des Wortes in der Bezeichnung betrachtet. Aus der induktiven Betrachtung von Bezeichnungen werden demnach deduktiv Schlüsse aus den Wortarten gezogen. Neben der Betrachtung der Wortarten werden aber auch Verfahren vorgestellt, welche die konkrete Ausprägung betrachten, um aus diesen Informationen für die Datenbeschaffung der anstehenden Simulationsstudie herzuleiten. Die Transformationsregeln, die für das fünfte Ziel hergeleitet werden, sind Gegenstand der Unterkapitel zu 3.6. Für diese Regeln wurde ein semantischer Abgleich zwischen dem Transformationsmodell und dem Simulationsmodell vollzogen, um die konkreten Transformationsregeln herzuleiten. Neben den zwei betrachteten Simulationsumgebungen wird in Kapitel 3.6.3 auf weitere Simulationsumgebungen eingegangen, die im Rahmen der Forschung untersucht wurden. Es werden jedoch keine Transformationsregeln zu diesen Simulationsumgebungen aufgeführt, sondern auf Probleme eingegangen, die eine Abbildung mit den Standardbausteinen verhindern.

Im Sinne der wissenschaftstheoretischen Grundposition nach Popper werden die einzelnen Regeln in Form von Hypothesen aufgestellt. Die Summe der einzelnen Transformationsregeln wird als Transformationstheorie bezeichnet. Aus einem syntaktisch korrekten Quellmodell soll durch die Transformationsregeln ein syntaktisch korrektes Zielmodell entstehen, dass den semantisch gleichen Ablauf aufweist. In Poppers Sinne kann aber nicht davon ausgegangen werden, dass die Regeln korrekt sind. Hierfür erfolgt in Kapitel 4 eine Evaluation, eine Falsifikation, der aufgestellten Regelbasis. Die dargestellte Falsifikation erfolgt für jede Regelbasis anhand von zwei ausgewählten Modellen. Wird nur der Kontext der Arbeit betrachtet, so kann nicht von einer strengen Falsifikation gesprochen wurde. Aus Platzgründen werden jedoch nicht mehr Modelle betrachtet, die im Zuge des Forschungsprozesses untersucht wurden. Die Unterkapitel 4.2 und 4.4 beinhalten die Prüfung der Transformationstheorien. Die Falsifikation der Normalisierung in Kapitel 4.3 kann diese Form jedoch nicht annehmen. Nach Anwendung der entsprechenden Regeln gibt es kein objektives Prüfkriterium, nach dem eine Aussage getroffen werden kann, ob die Regeln richtig ausgeführt wurden. Daher erfolgt nur eine Darlegung der Anwendung dieser Normalisierung. Die Evaluation schließt mit Kapitel 4.5, in dem die Ergebnisse der Evaluation diskutiert werden.

Im Kapitel 5 erfolgen die Zusammenfassung und die Darlegung der offenen Forschungsfragen. In Kapitel 5.1 wird aufgeführt, wie die zwei eingangs gestellten Forschungsfragen beantwortet wurden. Ebenfalls erfolgt eine Einschätzung, inwieweit die fünf gestellten Ziele erreicht wurden. Darauf aufbauend erfolgt eine Prüfung der Hypothesen. Dies umfasst die formulierten H1 bis H5 sowie eine Zusammenfassung der Prüfergebnisse der Falsifikationsversuch der Transformationstheorien. Die Arbeit schließt mit Kapitel 5.2, dass eine Untersuchung hinsichtlich resultierender Forschungsfragen aus den Erkenntnissen der Arbeit sowie Forschungsbedarf rund um das ProSiT Konzept beinhaltet.

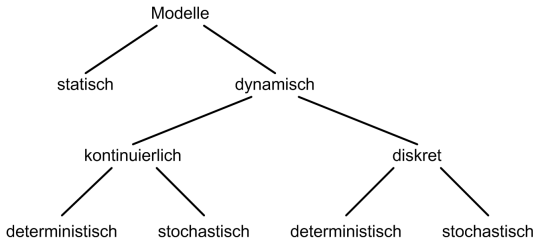
2 Theoretische Grundlagen zum Transformationsmodell

2.1 Der Modellbegriff und systemtheoretische Grundlagen

Zentraler Gegenstand dieser Arbeit sind Modelle. Diese werden nach dem in der Informatik häufigsten verwendeten Ansatz (Glinz 2008, S. 425) von Stachowiak (1974) definiert. Ein Modell zeichnet sich durch drei Merkmale aus, einem Abbildungs-, einem Verkürzungs- und einem pragmatischen Merkmal. Modelle sind nach dem Abbildungsmerkmal eine Repräsentation eines natürlichen oder künstlichen Originals. Nach dem Verkürzungsmerkmal repräsentieren Modelle nur Teile des Originals; mit den für Modellerschaffer oder -benutzer relevant erscheinenden Attributen. Das pragmatische Merkmal besagt, dass Modelle innerhalb eines Zeitintervalls für jemanden einen bestimmten Zweck erfüllen. Somit beantwortet das pragmatische Merkmal die Fragen für wen, wann und wozu, das Abbildungsmerkmal nach dem wo von und das Verkürzungsmerkmal nach dem Umfang des Modells (Stachowiak 1974, S. 131-133).

Im Kontext einer Simulation ist die Art der Zustandsübergänge bedeutsam, die in Abbildung 1 klassifiziert sind. Ein statisches Modell weist keine Zustandsänderungen auf und ist unabhängig von der Zeit. Bei einem dynamischen Modell hingegen sind die Modellzustände zeitabhängig. Bei einem kontinuierlichen Modell treten die Zustandsänderungen stetig auf, bei einem diskreten Modell sprunghaft, an konkreten Zeitpunkten. Dynamische Modelle können darüber hinaus in deterministisch und stochastisch unterteilt werden (Page 1991, S. 6).

Ein Geschäftsprozessmodell ist ein statisches Modell. In diesem werden zwar Zustandänderungen verbal beschrieben, dies betrifft jedoch Instanzen des Modells, die ausgeführt werden. Besonders wird dies durch die Notation der EPK deutlich. Ereignisse in der EPK stellen einen Zustand dar, der durch eine Funktion erzeugt wurde oder dieser vorangeht (Keller et al. 1992, S. 11). Nach dem eine Funktion ausgeführt wurde, hat die Instanz einen neuen Zustand. Details der Durchführung werden innerhalb eines Geschäftsprozessmodells nicht betrachtet.



(Page 1991, S. 6)

Abbildung 1: Modellklassifikation nach Art der Zustandsübergänge

Durch diese Beschreibung sind Zustandsänderung bei der Simulation von Geschäftsprozessmodellen sprunghaft und entsprechende Simulationsmodelle diskret (Tumay 1996, S. 97-98). Im Simulationsmodell können diese Zustandsänderungen entweder deterministisch oder stochastisch modelliert werden. Im Transformationsmodell erfolgt die Umwandlung des statischen Geschäftsprozessmodells in ein dynamisch diskretes Simulationsmodell.

Neben dem Modellbegriff werden Systeme betrachtet. Der Begriff des Systems wird in unterschiedlichsten wissenschaftlichen und praktischen Anwendungsgebieten verwendet. So wird eine Organisation als soziales System, ein Motor als technisches System oder eine Software als Anwendungssystem bezeichnet. Ebenfalls sind in der Betriebswirtschaftslehre verschiedene Systeme wie ein Zielsystem oder ein Produktionssystem anzutreffen. Selbst beim Modellbegriff handelt es sich um ein System (Forrester 1972, S. 16-17; Witte 1973, S. 2; Wilson 1990, S. 24-25; Lehner et al. 1995, S. 44-47).

Aus dieser Vielzahl an Systemen entstanden unterschiedliche Definitionen. Werden diese Definitionen gegenübergestellt, ergibt sich nach Witte (1973, S. 3) ein gemeinsamer Begriffsinhalt: „Ein System ist eine Gesamtheit von Elementen, zwischen denen Beziehungen bestehen".

Aus mathematischer Sicht beschreibt ein Mengenpaar (M, R) ein System. M stellt eine nicht leere Menge dar, die aus mindestens zwei Elementen besteht: $M_i; i = 1, \ldots, n, n \geq 2$. Die Menge R ist die Relationen zwischen den Elementen der Menge M, die in der Anzahl n vorkommt, wobei $n \geq 1$ gilt (Witte 1973, S. 4). Diese Definition wird der Arbeit zugrunde gelegt.

Ein System grenzt Elemente von der Umwelt ab (Känel 1972, S. 42-43; Krieger 1996, S. 16). Zu unterscheiden sind offene und geschlossene Systeme. Liegt ein Austausch mit der Umwelt vor, ist es ein offenes System, sei es Energie, Materie oder Information; ohne diesen ein geschlossenes. Diese Unterscheidung ist abhängig vom Standpunkt des Betrachters beziehungsweise der Abgrenzung (Forrester 1972, S. 15; Vetter 1994, S. 48-49; Lehner et al. 1995, S. 49-50; Krieger 1996, S. 38-39).

Die einzelnen Elemente eines Systems können über Attribute verfügen. Diese können verschiedenen Zuständen aufweisen. Bei einem geschlossenen System werden diese Zustände lediglich durch andere Elemente im System beeinflusst; die Zustände in offenen Systemen zusätzlich von der Umwelt. Dies erfolgt durch den Input, den das System aus der Umwelt erfährt. Der an die Umwelt abgegebene Output hat keinen Einfluss auf das System (Forrester 1972, S. 15; Vetter 1994, S. 48-49; Krieger 1996, S. 38-39). Geschäftsprozesse sind durch mindestens ein Start- und Endereignis gekennzeichnet. Diese stellen eine Verbindung zur Umwelt dar, entsprechend handelt es sich um offene Systeme.

Ausgehend von der Betrachtung eines Systems, kann dieses sich in einem Übersystem befinden, oder selbst Untersysteme beinhalten. Ein Übersystem umschließt das betrachtete System, während ein Untersystem vollständig in einem System enthalten ist. Das oberste System in dieser Hierarchie wird selbst wieder von der Umwelt umschlossen (Vetter 1994, S. 55-63; Wilson 1990, S. 34).

Eine weitere Unterscheidung von Systemen erfolgt hinsichtlich zeitlicher Änderungen der Eingangs- und Ausgangsgrößen sowie der Zustände der Elemente des Systems. Zu unterscheiden sind, ähnlich der Modelle, zeitkontinulierliche und zeitdiskrete Systeme (Känel 1972, S. 50-51; Friedrich 1984, S. 138). Bei Geschäftsprozessmodellen handelt es sich um zeitdiskrete Systeme, da Änderungen nur zu bestimmten Zeitpunkten im Modell erfolgen.

2.2 Zum Begriff der Prozess- und Simulationsmodelle

Zwei Arten von Modellen werden in dieser Arbeit besonders betrachtet: Prozess- und Simulationsmodelle. Bei einem Prozessmodell handelt es sich aus begrifflichen Gründen um das Modell eines Prozesses. Mit dem Begriff Prozess ist im Rahmen der Wirtschaftsinformatik der Begriff Geschäftsprozess verbunden. Wie Becker und Schütte (2004, S. 107) aufführen, kann es sich bei einem Geschäftsprozess und einem Prozess, aus sprachlicher Sicht, nicht um das Gleiche handeln und somit können beide Begriffe nicht synonym verwendet werden, da ein Geschäftsprozess eine begriffliche Spezialisierung des Prozesses ist. Für die Abgrenzung dieser beiden Begriffe wird eine volkswirtschaftlich orientierte Abgrenzung herangezogen. Daraus resultiert, dass ein Geschäftsprozess ein Prozess ist, der in einem rechtlich verselbstständigten Unternehmen (oder beim Staat) durchgeführt wird.

In der Literatur gibt es keine einheitliche Definition für den Begriff des Prozesses. Nachfolgend werden ausgewählte Definitionen für den Begriff aufgeführt und aus diesen die Definition für diese Arbeit abgeleitet.

Die ISO 9000:2005 (DIN 2005, S. 23) definiert einen Prozess als „Satz von in Wechselbeziehungen oder Wechselwirkung stehenden Tätigkeiten, der Eingaben in Ergebnisse umwandelt." Rosemann (1996, S. 9) hingegen definiert einen Prozess mit: „Ein Prozess stellt die inhaltlich abgeschlossene, zeitliche und sachlogische Abfolge der Funktionen dar, die zur Bearbeitung eines betriebswirtschaftlich relevanten Objekts ausgeführt werden." Allweyer (2005, S. 4) sieht in einem Geschäftsprozess „[...] die zur Erstellung von Produkten und Leistungen erforderlichen betrieblichen Abläufe." Gadatsch (2010, S. 41) verwendet als Definition für einen Geschäftsprozess "[...] eine zielgerichtete, zeitlich-logische Abfolge von Aufgaben, die arbeitsteilig von mehreren Organisationen oder Organisationseinheiten unter Nutzung von Informations- und Kommunikationstechnologien ausgeführt werden können." Nach Davenport (1993, S. 5), eine im englischsprachigen Raum häufig verwendete Definition, ist ein Prozess eine spezifische Reihenfolge von Arbeitsvorgängen über Zeit und Orte, mit einem Anfang, einem Ende und klar identifizierten Eingaben und Ausgaben, eine Struktur für Tätigkeiten.

Aus den Definitionen geht hervor, dass zum einen eine Abfolge vorliegt, zum anderen etwas bearbeitet wird. Im Rahmen dieser Arbeit werden diese Aspekte besonders berücksichtigt und eine eigene Definition für den Begriff des Prozesses hergeleitet, welche die auszuführende Arbeit in den Mittelpunkt stellt. Ein Prozess wird definiert als die zeitlich-logische Abfolge von Arbeitsschritten. Dementsprechend ist nach der volkswirtschaftlichen Abgrenzung ein Geschäftsprozess eine zeitlich-logische Abfolge von Arbeitsschritten in einem rechtlich verselbstständigten Unternehmen.

Kernelement der Definition des Prozesses ist der Begriff Arbeitsschritt. Dieser verdeutlicht die arbeitsteilige Ausführung eines Prozesses. Er wird definiert als die bewusste Auseinandersetzung eines Subjekts mit der Umwelt. Hierbei wird nicht unterschieden, ob es sich bei dem Subjekt um einen Menschen, ein Tier oder eine Maschine handelt. Aus dieser Definition geht hervor, dass ein Arbeitsschritt eine Kombination von einer Tätigkeit, einem Objekt und ein oder mehrerer Ressourcen ist.

Unter einer Tätigkeit kann ein Handgriff oder eine in sich abgeschlossene Gruppe von Handgriffen verstanden werden. Beispiele für eine Tätigkeit sind „anlegen", „erzeugen", „bearbeiten", „prüfen", „genehmigen" aber auch „schneiden", „zubereiten", „fräsen" oder „bohren". Unter einem Objekt wird der Gegenstand einer Bearbeitung beziehungsweise der Tätigkeit verstanden. Ressourcen hingegen werden als Mittel für die Ausführung einer Tätigkeit herangezogen. Aus sprachlicher Sicht kann die Tätigkeit eines Arbeitsschritts nur mit einem transitiven Verb verbal formuliert werden. Nur diese Verben setzen ein Objekt voraus. Eine mit einem intransitiven Verb formulierte Tätigkeit, kann daher nicht Teil eines Arbeitsschrittes sein, da Ressourcen die Tätigkeit nicht an einem Objekt ausführen.

Prozessmodelle können auf verschiedene Arten modelliert werden (Allweyer 2005, S. 130): Beschreibung als Text, tabellarische Darstellung, grafische Darstellung ohne Verwendung einer bestimmten Notation und grafische Modellierung mithilfe einer definierten Notation. Im Rahmen der Arbeit werden nur Prozessmodelle, die mithilfe einer definierten Notation modelliert werden, betrachtet. Diese Art von Modellen wird auch als semiformale Modelle bezeichnet (Staud 2006, S. 59).

Bezogen auf das Kerngeschäft einer Organisation können Geschäftsprozesse in Führungs-, Kern- und Unterstützungsprozesse unterteilt werden[5]. Diese Differenzierung wird als Prozesstyp bezeichnet. Führungsprozesse umfassen die Planung und Steuerung sowie die strategischen Elemente einer Organisation. Kernprozesse beschreiben die Leistungserstellung – Wertschöpfung – der Organisation. Unterstützungsprozesse tragen zu dieser einen geringen Teil bei und sorgen für einen reibungslosen Ablauf des operativen Geschäfts (Allweyer 2005, S. 73-76; Gadatsch 2010, S. 44-45).

Zur Abgrenzung von Prozessen kann nach Becker und Schütte (2004, S. 107) der Objektbezug verwendet werden (siehe auch Frank und van Laak 2003, S. 68). Der Geschäftsprozess der Kundenauftragsabwicklung hat als Objekt den Kundenauftrag, der des Wareneingangs die Ware. Gegenstand eines Prozesses ist immer ein Objekt, das als Prozessobjekt bezeichnet wird. Das Objekt eines Arbeitsschritts muss mit dem Prozessobjekt nicht übereinstimmen.

Die Modellierung von Geschäftsprozessen erfolgt primär in einer semiformalen festgeschriebenen Notation (Allweyer 2004, S. 130; Gadatsch 2010, S. 71). Ziele, die mit der Modellierung von Geschäftsprozessen erreicht werden sollen, sind unter anderem die Erfassung und Dokumentation der betrieblichen Abläufe, eine Schwachstellenanalyse der bestehenden Prozesslandschaft, eine Anforderungsdefinition für neue Informationssysteme, die Auswahl und Einführung einer Standardsoftware oder der Aufbau eines Unternehmensprozessmodells (Gadatsch 2008, S. 119). Diese Ziele repräsentieren aus Sicht des Modellbegriffs das pragmatische Merkmal eines Geschäftsprozessmodells und bestimmten, welche Elemente im Geschäftsprozessmodell berücksichtigt werden. Für eine Simulation eines Geschäftsprozesses beziehungsweise dessen Model können die Informationen ausgeblendet werden, die aufgrund einer anderen

5 Begrifflich erfolgt in der Literatur bei Führungs-, Kern- und Unterstützungsprozessen keine Unterscheidung zwischen den Begriffen Prozess und Geschäftsprozess. Bezogen auf die Abgrenzung von Prozess und Geschäftsprozess kann die Unterteilung nur bei Geschäftsprozessen erfolgen.

Zielstellung enthalten sind. Umgekehrt bedeutet dies, dass nicht davon ausgegangen werden kann, dass ein Geschäftsprozessmodell alle für eine Simulation notwendigen Daten beinhaltet, wenn dieses für einen andern Anwendungszweck modelliert wurde.

Der zweite wichtige Begriff in dieser Arbeit ist Simulation und damit zusammenhängende Simulationsmodelle. In der Literatur wird der Begriff Simulation, wie der Prozessbegriff, unterschiedlich definiert (Witte 1973 S. 18; Shannon 1992, S. 65; Lowery 1998, S. 31; Allweyer 2005, S. 244; Gadatsch 2010, S. 216; Rossetti 2010, S. 2), jedoch wird im deutschsprachigen Raum, insbesondere in der Ingenieurwissenschaft, die Definition der VDI am häufigsten verwendet. Der VDI (1995a, S. 14) definiert Simulation als „[...] ein Verfahren zur Nachbildung eines Systems mit seinen dynamischen Prozessen in einem experimentierbaren Modell, um zu Erkenntnissen zu gelangen, die auf die Wirklichkeit übertragbar sind." Im weiteren Sinne wird unter Simulation das Vorbereiten, Durchführen und Auswerten gezielter Experimente mit einem Simulationsmodell verstanden."

Der Ansatz mittels Transformationsmodell stellt ein Verfahren zur Nachbildung eines Systems dar. Die dynamischen Prozesse ergeben sich aus den Geschäftsprozessmodellen, die als Grundlage verwendet werden. Das experimentierbare Modell liegt vor, wenn das Transformationsmodell in eine Simulationsmodell überführt wurde. Der weitere Sinn, unter dem Simulation verstanden wird, ist nur zum Teil abgedeckt. Aus diesem wird nur das Vorbereiten vom Transformationsmodell Ansatz abgedeckt. Die Durchführung und Auswertung von Experimenten, die mit dem Simulationsmodell ausgeführt werden, sind aus der Betrachtung ausgeschlossen. Erkenntnisse, die durch eine Simulation eines Geschäftsprozesses gewonnen werden sollten, betreffen unter anderem Durchlaufzeiten, Wartezeiten, Entwicklungen von Warteschlangen, Auslastungen von Ressourcen sowie entstehende Kosten (Allweyer 2005, S. 257). Mittels animierter Simulationen lassen sich Abläufe visuell validieren oder das Verhalten von vernetzen Geschäftsprozessen untersuchen (Gadatsch 2008, S. 229).

2.3 Notationen von Prozessmodellen

Im Rahmen dieser Arbeit werden drei Prozessmodellierungsnotationen als Quellmodelle betrachtet. Auf diese Notationen wird in Kapitel 3 für die Aufstellung der Transformationsregeln zurückgegriffen. Betrachtet werden die erweiterten Ereignisgesteuerten Prozesskette (eEPK), die Business Process Model and Notation (BPMN) und das Aktivitätsdiagramm der Unified Modelling Notation (UML). Diese drei Notationen gehören nach Gadatsch (2010, S. 71) zu den mit am häufigsten genutzten Notationen im deutschsprachigem Raum.

Ein weiterer Grund für die Wahl der eEPK liegt in der Verwendung in den grundlegenden Lehrwerken zum Geschäftsprozessmanagement (Scheer 1997; Becker und Schütte 2004; Allweyer 2005; Gadatsch 2010). Für die BPMN spricht die Standardisierung im Rahmen der Object Management Group (OMG 2011). Die BPMN (OMG 2011, S. 27-41) bietet mehr Modellierungselemente als die eEPK (Gadatsch 2010, S. 87) und erlaubt eine einfacherer Modellierung. Schwierigkeiten mit Zwischenereignissen in der eEPK, die als trivial angesehen werden, oder mit dem XOR und OR Operatoren nach einem einzelnen Ereignis (Overhage et al. 2011, S. 751) sind in der BPMN nicht vorhanden. Das UML Aktivitätsdiagramm als dritte betrachtete Notation ist ebenfalls ein Standard bei der OMG (2010). Die UML ist für die Modellierung von Informationssystemen konzipiert und hat nach Störrle (2006, S. 184) in dieser Domäne Vorteile gegenüber der eEPK. Verwendet wird die Notation aber auch bei der Modellierung von Geschäftsprozessen (Oestereich 2004, S. 13; Staud 2006, S. 310; Gadatsch 2010, S. 71; Funk et al. 2010, S. 33). Hinsichtlich der Anzahl der Modellierungselemente ist das UML Aktivitätsdiagramm ebenfalls der eEPK überlegen. Alle Konstrukte der eEPK lassen sich mit dem UML Aktivitätsdiagramm abbilden; umgekehrt aber nicht (Störrle 2006, S. 183). Hinsichtlich der Modellerstellung sei aber weder die eEPK noch das UML Aktivitätsdiagramm überlegen (Overhage et al. 2011, S. 753). Petri-Netze werden als Quellmodelle im Rahmen dieser Arbeit nicht betrachtet. Grund hierfür liegt in der geringen Verwendung zur

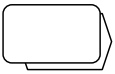
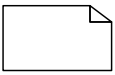
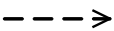
Modellierung von Geschäftsprozessen (Gadatsch 2010, S. 71). Gründe könnten darin liegen, dass eEPKs im Vergleich zu Petri-Netzen verständlicher sind und daher bevorzugt werden (Sarshar und Loos 2005, S. 437)

In den drei nachfolgenden Unterkapiteln werden die drei ausgewählten Notationen kurz beschrieben, mit der Aufführung der betrachteten Elemente und dessen Beziehungen.

2.3.1 Erweiterte Ereignisgesteuerte Prozesskette

Für die Beschreibung der Notation der eEPK wird das von Mendling und Nüttgens (2006) vorgeschlagene Austauschformat, die EPC markup language (EPML), herangezogen. Bei EPML handelt es sich um ein XML Format, das zum einen die Logik, zum anderen die grafische Darstellung einer eEPK beschreibt. Neben den Elementen der EPK, der Funktion, dem Ereignis und den Operatoren AND, OR und XOR umfasst es drei erweiternde Elemente, ein Anwendungssystem (application), einen Teilnehmer (participant) und Daten (dataField). Ebenfalls berücksichtigt das Austauschformat die Prozess-Schnittstelle (processInterface). Die Elemente die in der eEPK Notation des EPML Format unterstützt werden, sind in Tabelle 1 dargestellt.

Tabelle 1: Elemente einer eEPK im EPML Format

Grafische Darstellung	Element	Beschreibung
	Funktion (Function)	Beschreibung einer Tätigkeit, welche eine Transformation zwischen einem Input- und einem Output-Zustand darstellt.
	Ereignis (Event)	Beschreibung eines eingetretenen betriebswirtschaftlichen Zustandes.
	Prozess-Schnittstelle (ProcessInterface)	Verweis auf einen vorhergehenden oder weiterführenden Prozess.
	Logischer Operator „und“ (And)	Konjunktion A und B
	Logischer Operator „inklusives oder“ (Or)	Adjunktion Entweder A oder B oder A und B
	Logischer Operator „exklusives oder“ (Xor)	Disjunktion A oder B, aber nicht A und B
	Stelle (Participant)	Beschreibung der ausführenden Einheit einer Funktion
	Anwendungssystem (Application)	Beschreibung eines Anwendungssystems, welches eine Funktion ausführt oder für die Ausführung von einer Stelle herangezogen wird.
	Informationsobjekt (DataField)	Beschreibung einer Information, die für die für Ausführung einer Funktion benötigt wird oder von dieser erzeugt wird.
	Kontrollfluss (Arc)	Sequenzielle Folge zwischen zwei Elementen.
	Relation (Relation)	Beziehung zwischen zwei Elementen.

Die Beziehungen zwischen den Elementen werden im EPML Format mit dem Kontrollfluss und der Relation beschrieben. In diesen Verbindungselementen wird angegeben, welches Element das Quell-

und welches das Zielelement ist. Die Elemente beinhalten im EPML Format keine Information, mit welchen andern Elementen diese verknüpft sind (Mendling und Nüttgens 2006, S. 253). Für die klarere Formulierung der Transformationsregeln werden die Beziehungen zwischen den Elementen als Attribut der Elemente aufgefasst.

Die Verknüpfungsmöglichkeiten der Elemente aus Tabelle 1 sind in Abbildung 2 dargestellt. Die Abbildung orientiert sich an der Darstellung der Verknüpfungsmöglichkeiten einer EPK bei Mendling und Nüttgens (2003a, S. 22). Graue Zellen stellen einen Funktions-Ereignis-Kontrollfluss (FE), weiße Zellen mit Pfeil einen Ereignis-Funktions-Kontrollfluss (EF) dar. Diese Unterscheidung begründet sich im Wechsel von Funktion und Ereignis. Das „S" bei den Operatoren steht für einen Split, das „J" für einen Join.

TO / FROM	E_S	E_{Int}	E_E	F	P_S	P_E	AND_{EFS}	AND_{EFJ}	AND_{FES}	AND_{FEJ}	OR_{EFJ}	OR_{FES}	OR_{FEJ}	XOR_{EFJ}	XOR_{FES}	XOR_{FEJ}
E_S				→		→	→	→			→			→		
E_{Int}				→		→	→	→			→			→		
E_E																
F		→	→						→	→		→	→		→	→
P_S		→	→													
P_E																
AND_{EFS}				→			→	→			→			→		
AND_{EFJ}				→			→	→			→			→		
AND_{FES}		→	→						→	→		→	→		→	→
AND_{FEJ}		→	→						→	→		→	→		→	→
OR_{EFJ}				→			→	→			→			→		
OR_{FES}		→	→						→	→		→	→		→	→
OR_{FEJ}		→	→						→	→		→	→		→	→
XOR_{EFJ}				→			→	→			→			→		
XOR_{FES}		→	→						→	→		→	→		→	→
XOR_{FEJ}		→	→						→	→		→	→		→	→

In Anlehnung an (Mendling und Nüttgens 2003a, S. 22)

Abbildung 2: Verknüpfungsmöglichkeiten der EPK

Neben diesen Verknüpfungsmöglichkeiten kann auch die Prozessschnittstelle – als Vorgänger eines Startereignisses oder als Nachfolger

eines Endereignisses – verwendet werden, um auf einen anderen Prozess der gleichen Hierarchiestufe zu verweisen. Eine Funktion kann, neben ihrer Repräsentation eines Arbeitsschrittes, auch als Verweis auf einen Unterprozess eingesetzt werden. Gekennzeichnet wird dies über das Attribut „toProcess“ im EPML Format (Mendling und Nüttgens 2006, S. 255). Diese drei Verknüpfungsmöglichkeiten stellt Abbildung 3 dar.

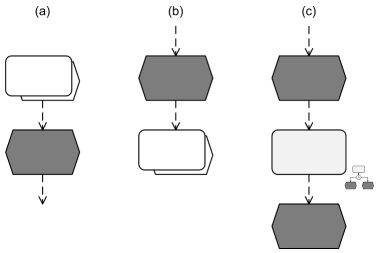


Abbildung 3: Prozessschnittstelle in der EPK

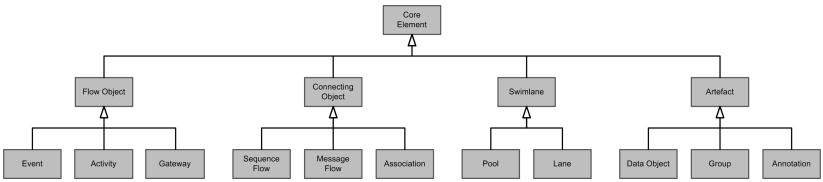
2.3.2 Business Process Model and Notation

Die Business Process Model and Notation, in der Version 2.0, ist heute eine Spezifikation der OMG (2011, S. 1). Für die Konzeption dieser Notation wurden bestehende Modellierungssprachen untersucht und aus diesen, nach Angaben der Spezifikation, die besten Ideen entnommen. Darin enthalten waren unter anderem das UML Aktivitätsdiagramm (OMG 2010) und die EPK (Keller et al. 1992); BPMN beinhaltet daher Elemente aus unterschiedlichen Modellierungssprachen.

Die Modellierungskonzepte der BPMN beschränken sich auf die reine Modellierung von Geschäftsprozessen. Die Modelle unterscheiden sich in: Private nicht-ausführbare, private ausführbare und öffentliche Geschäftsprozesse, Choreographien und Kollaborationen (OMG 2011, S. 22-24). Choreographie-Diagramme dienen der Darstellung von Geschäftskontakte zwischen zwei oder mehreren Unternehmen (OMG 2011, S. 315). Während bei diesen eigene Notationselemente vorhanden sind, verwenden Kollaborationen die Elemente des Prozesses.

Aufgrund des Kernelements des Arbeitsschritts – mit der Notwendigkeit von Ressourcen – werden diese beiden Modelle nicht betrachtet. Ebenfalls werden öffentliche Prozesse[6] nicht betrachtet, da nur Aktivitäten mit Interaktionen eines Geschäftspartner berücksichtigt werden. Die beiden privaten Prozesse unterscheiden sich in der Menge der hinterlegten Informationen; ausführbare Prozesse haben Daten, die für eine Workflow-Engine notwendig sind. Aus simulationstechnischer Sicht sind diese Daten aber nicht notwendig. Beide privaten Prozesse werden daher als Quellmodelle herangezogen. Um eine leichtere Lesbarkeit zu gewährleisten, bezieht sich im Folgenden der Begriff BPMN nur auf die Spezifikationen der privaten Prozesse (OMG 2011, S. 145-314).

Die Elemente der privaten Prozesse werden in Kernelemente zusammengefasst (OMG 2011, S. 27-30). Mit diesen Kernelementen können die Beziehungen zwischen den Elementen beschrieben werden, beispielsweise das Kernelement Gateway anstelle des exklusiven oder inklusiven Gateways. Die Kernelemente sind in Abbildung 4 dargestellt und werden in Tabelle 2 herangezogen, um die Elemente einzuordnen[7].



Nach (OMG 2011, S. 27-28)

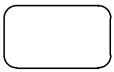
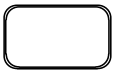
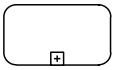
Abbildung 4: BPMN Kernelemente

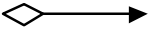
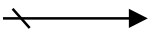
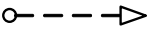
Eine weiterführende Erläuterung der nachfolgend aufgeführten Elemente ist in den Spezifikationen der BPMN (OMG 2011) sowie in den Büchern von Allweyer (2009) und Freund et al. (2010) vorzufinden.

6 Ein anschauliches Beispiel für die Unterschiede zwischen privaten und öffentlichen Prozessen ist in Allweyer (2009, S. 55-58) zu finden.

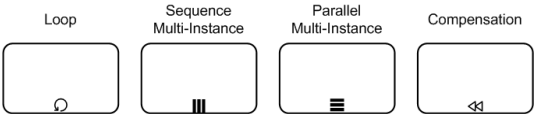
7 Das Kernelement wird in das Spalte Element in Klammern angegeben.

Tabelle 2: Betrachtete Elemente des BPMN Process

Grafische Darstellung	Element	Beschreibung
	Aufgabe (Activity)	Eine Aufgabe ist ein elementarer Arbeitsschritt zur Verrichtung einer Tätigkeit.
	Transaktion (Activity)	Eine Transaktion fasst eine Summe von logisch zusammengehörenden Aufgaben zusammen.
	Teilprozess (Activity)	Ein Teilprozess stellt einen in sich geschlossenen Prozess dar.
	Aufrufaktivität (Activity)	Eine Aufrufaktivität ist eine Referenz auf eine Aufgabe oder einen Teilprozess. Das referierte Element wird durch diese aufgerufen.
	Exklusives Gateway (Gateway)	Ein exklusives Gateway stellt ein logisches exklusives Oder dar, bei dem nur ein Pfad ausgeführt wird.
	Ereignis-basiertes Gateway (Gateway)	Bei einem Ereignis-basierten Gateway wird der auszuführende Prozesspfad durch ein eingetretenes Ereignis bestimmt.
	Paralleles Gateway (Gateway)	Ein paralleles Gateway stellt ein logisches Und dar, bei dem alle Pfade ausgeführt werden.
	Inklusives Gateway (Gateway)	Ein inklusives Gateway stellt ein logisches inklusives Oder dar. Ein teilendes Gateway führt ein oder mehrere nachfolgende Prozesspfade aus, die bei einem zusammenführenden abgeschlossen sein müssen, um den Prozess fortzusetzen.
	Komplexes Gateway (Gateway)	Ein komplexes Gateway bietet weitreichende Möglichkeiten nachfolgende Prozesspfade zu wählen, beispielsweise zwei von drei; eine definierte Syntax hierfür liegt aber nicht vor (OMG 2011, S. 295-297).
	Exklusives Ereignisbasiertes Gateway	Ein exklusives ereignisbasiertes Gateway ist einen Startpunkt im Prozess; tritt nur ein Ereignis ein, wird der Prozess gestartet.

Grafische Darstellung	Element	Beschreibung
	Paralleles Ereignis-basiertes Gateway	Ein paralleles ereignisbasiertes Gateway ist ebenfalls der Startpunkt eines Prozesses; bei dem alle Ereignisse eintreten müssen.
	Startereignis (Event)	Ein Startereignis stellt einen Startpunkt des Prozesses dar. Ab diesem Ereignis wird der Ablauf betrachtet.
	Zwischenereignis (Event)	Ein Zwischenereignis wird während eines Prozesses verwendet, um einen bestimmten Sachverhalt darzustellen, beispielsweise das Senden einer Nachricht oder das Warten auf ein Signal.
	Endereignis (Event)	Ein Endereignis stellt einen Endpunkt des Prozesses dar. Ab diesem Ereignis wird der Ablauf nicht mehr betrachtet.
	Pool	Ein Pool gruppiert Lanes. Innerhalb eines Pools werden Sequenzflüsse, poolübergreifend Nachrichtenflüsse verwendet.
	Lane	Eine Lane kategorisieren Aktivitäten innerhalb eines Prozesses. Wie Lanes zu verwenden sind, überlässt BPMN dem Modellierer (OMG 2011, S. 306).
	Sequenzfluss	Ein Sequenzfluss stellt eine sequenzielle Abfolge zwischen zwei Elementen dar.
	Bedingter Fluss	Ein bedingter Fluss kann anstelle eines Gateways verwendet werden, um den weiteren Prozessverlauf zu definieren.
	Standardfluss	Der Standardfluss kann nach Gateways und Aktivitäten, gemeinsam mit Sequenz- und bedingten Flüssen, verwendet werden bei denen eine Entscheidung über nachfolgende Prozesspfade getroffen wird.
	Nachrichtenfluss	Der Nachrichtenfluss verknüpft Elemente poolübergreifend und stellt eine Interaktion dar.

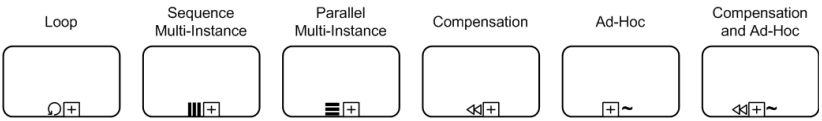
Im Gegensatz zur Funktion der EPK kann die Aufgabe im BPMN weiter spezifiziert werden, mit dem Typ der Aufgabe und einem Marker. Der Typ repräsentiert das innere Verhalten (OMG 2011, S. 158). Für das Transformationsmodell ist das innere Verhalten[8] aber nicht relevant; entsprechend wird dieses nicht berücksichtigt. Marker hingegen beeinflussen den Ablauf des Geschäftsprozesses. Die vier Marker für eine Aufgabe (OMG 2011, S. 157) sind in Abbildung 5 dargestellt.



Nach OMG (2011, S. 157)

Abbildung 5: Marker bei einer Aufgabe in BPMN

Eine Aufgabe mit „Loop" Marker wird so lange durchgeführt, bis ein Abbruchkriterium erfüllt ist. Die Anzahl der Schleifen sind zu Beginn der Aufgabe nicht festgelegt, im Gegensatz zum „Sequence Multi-Instance" Marker, bei dem die Anzahl am Anfang definiert wird. Der „Parallel Multi-Instance" Marker legt eine mehrfache parallele Ausführung der Aufgabe fest. Nur im Rahmen von Ausnahmebehandlungen wird der „Compensation" Marker verwendet, um beispielsweise Korrekturen bei eingetretenen Fehlern vorzunehmen.



Nach OMG (2011, S. 175)

Abbildung 6: Marker beim Teilprozess in BPMN

Analog gibt es diese vier Marker für einen Teilprozess (Abbildung 6) sowie zwei weitere (OMG 2011, S. 175). Bei einem „Ad-Hoc" Teilprozess ist die Reihenfolge des Prozessablaufs nicht festgelegt. Der sechste Marker „Compensation und Ad-Hoc" kombiniert beide und legte keine Reihenfolge für Korrekturen fest.

8 Eine Übersicht über die vorhandenen Aufgabentypen ist in OMG (2011, S. 158-165) vorzufinden.

Neben den von der Aktivität[9] abgeleiteten Elementen können auch Ereignisse mit Markern versehen werden. Eine Gesamtübersicht der Marker bietet Abbildung 7.

Types	Start			Intermediate				End
	Top-Level	Event Sub-Process *Interrupting*	Event Sub-Process *Non-Interrupting*	Catching	Boundary *Interrupting*	Boundary *Non-Interrupting*	Throwing	
None								
Message								
Timer								
Error								
Escalation								
Cancel								
Compensation								
Conditional								
Link								
Signal								
Terminate								
Multiple								
Parallel Multiple								

OMG (2011, S. 261-262)

Abbildung 7: Marker bei Ereignissen in BPMN

9 Die Aufrufaktivität sowie Transaktion können ebenfalls mit einem Marker versehen werden (OMG 2011, S. 179, 184). Diese entsprechend denen der Aufgabe (Abbildung 5) und des Teilprozesses (Abbildung 6).

Neben alleinstehenden Zwischenereignissen, die in Tabelle 2 aufgeführt sind, können diese auch an Aktivitäten angehängt werden, die entweder eine Aktivität abbrechen (Boundary Interrupting) oder einen weiteren Prozesspfad beim Eintreten auslösen (Boundary Non-Interrupting). Ein Beispiel für angeheftete Ereignisse ist in Abbildung 8 dargestellt.

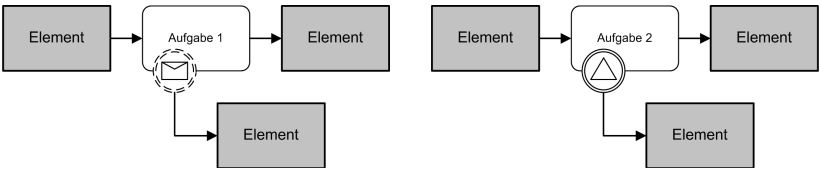


Abbildung 8: Angeheftete Ereignisse an Aufgaben in BPMN

Diese beiden Ereignistypen sind in der BPMN Notation auch als Startereignisse spezifiziert (OMG 2011, S. 242) und werden im Rahmen eines Ereignis-Teilprozesses verwendet. Der Ereignis-Teilprozess kann nur in einem Teilprozess integriert werden und nur während der Ausführung einer Instanz gestartet werden. Keine der betrachteten Simulationsumgebungen unterstützt dieses Konstrukt, da alle Instanzen in einem Modell ablaufen. In einer Workflow-Engine wäre es hingegen möglich, da jede Instanz separat behandelt wird. Da Instanzen nicht unterschieden werden können, ist der Ereignis-Teilprozess sowie dessen beide Startereignisse von der Betrachtung ausgeschlossen.

Eine Erläuterung der Marker ist in den Spezifikationen vorzufinden (OMG 2011, S. 240-257). Nachfolgend wird nur auf die Marker eingegangen, die für die Simulation des BPMN Prozesses eine besondere Rolle spielen. Das Nachrichten („Message“) Ereignis kann zur Modellierung eines Nachrichtenflusses zwischen zwei Pools verwendet werden, entweder als sendendes oder empfangendes Ereignis. Das „Timer“ Ereignis stellt entweder einen bestimmten Zeitpunkt oder eine Zeitspanne dar; wird aber immer als relative Zeitspanne interpretiert, da nur diese in den Simulationsumgebungen umsetzbar ist. Ist ein Prozess grafisch nicht verständlich darstellbar, bietet der „Link“ Marker eine alternative Darstellung zum Sequenzfluss. Das sendende Ereignis stellt den Sprungpunkt dar und das empfangende den Zielpunkt. Der letzte hervorgehobene („Terminate“)

Marker, der nur bei einem Endereignis vorkommt, beendet eine laufende Instanz, ohne Kompensationen auszuführen. Andere Endereignisse beenden im Gegensatz dazu nur den eigenen Prozesspfad.

Ein Kernelement aus Abbildung 4 wird in Tabelle 3 nicht berücksichtigt, das Artefakte. Den Artefakten werden Datenelemente (OMG 2011, S. 203), Gruppen (OMG 2011, S. 68) und Anmerkungen (OMG 2011, S. 71) zugeordnet. Diese unterstützen das Verständnis für den Betrachter des Modells, haben aber keinen Einfluss auf die Simulation eines Geschäftsprozesses und sind von der Betrachtung ausgeschlossen.

Tabelle 3: Verbindungsmöglichkeiten mit dem Sequenzfluss in BPMN

Von \ Nach						
		→	→	→	→	→
		→	→	→	→	→
		→	→	→	→	→
		→	→	→	→	→
					→	
					→	
		→	→	→	→	→
						

Nach OMG (2011, S. 42-43)

Durch Wegfall der Artefakte werden Assoziationen nicht benötigt, um diese mit Flussobjekten zu verknüpfen. Entsprechend sind in Tabelle 3 die Verknüpfungsmöglichkeiten mit dem Sequenzfluss und in Tabelle 4 mit dem Nachrichtenfluss aufgeführt. Transaktion und Aufrufaktivität werden durch die Aufgabe und den Teilprozess indirekt berücksichtigt.

Tabelle 4: Verbindungsmöglichkeiten mit dem Nachrichtenfluss in BPMN

Von \ Nach						
						
	→	→	→	→	→	
	→	→	→	→	→	
	→	→	→	→	→	
	→	→	→	→	→	
	→	→	→	→	→	

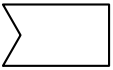
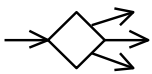
Nach OMG (2011, S. 44)

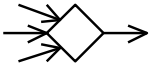
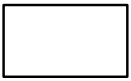
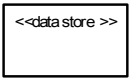
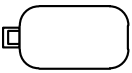
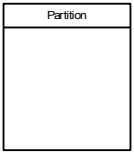
2.3.3 UML Aktivitätsdiagramm

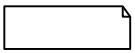
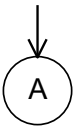
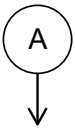
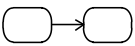
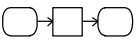
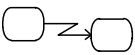
Das UML Aktivitätsdiagramm ist Teil der Unified Modeling Language (UML) Spezifikation der OMG (2010). Anwendungsschwerpunkt ist die objektorientierte Softwareentwicklung. Hierfür spezifiziert die UML verschiedene Diagrammtypen, von denen einer das UML Aktivitätsdiagramm (UML AD) ist. Dieses stellt ein Ablaufdiagramm mit Knoten und Kanten dar (Arlow und Neustadt 2005, S. 283) und orientiert sich an Petri-Netzen (OMG 2010, S. 333). Im Gegensatz zur eEPK wird aber der Objektfluss und Kontrollfluss als Kante getrennt modelliert (Staud 2010, S. 200). Die Knoten im UML Aktivitätsdiagramm unter-

liegen einer Dreiteilung in Aktions-, Objekt- und Kontrollknoten (Oestereich 2006, S. 302). Die einzelnen Elemente des UML Aktivitätsdiagramms sind in der nachfolgenden Tabelle 5 aufgeführt. Beim UML Aktivitätsdiagramm ist zu beachten, dass der modellierte Ablauf als Aktivität bezeichnet wird (Oesterle 2006, S. 307).

Tabelle 5: Elemente des UML Aktivitätsdiagramms

Grafische Darstellung	Element	Beschreibung
	Startknoten [InitialNode] (Startknoten)	Ein Startknoten repräsentiert den Anfang einer Aktivität. Beginnend am Startknoten durchlaufen Marken die Aktivität.
	Flussende [FlowFinal] (Endknoten)	Ein Flussende stellt das Ende der Ausführung eines Flusses. Erreicht eine Marke diesen Knoten wird diese entfernt.
	Aktivitätsende [ActivityFinal] (Endknoten)	Erreicht eine Marke das Aktivitätsende, wird die komplette Aktivität beendet, unabhängig davon ob noch andere Aktionen ausgeführt werden.
	Aktion [Action] (Aktionsknoten)	Eine Aktion stellt einen Arbeitsschritt dar, der nicht weiter unterteilt wird.
	Signalsendeaktion [SendSignalAction] (Aktionsknoten)	Das von einer Signalsendeaktion gesendete Signal kann von einer Signalereignisaktion empfangen werden. Mit diesen Elementen ist eine Synchronisation nebenläufige möglich (Oesterle 2006, S. 314).
	Signalereignisaktion [AcceptEventAction] (Aktionsknoten)	Die Signalereignisaktion empfängt ein Signal und setzt daraufhin die Verarbeitung der Aktivität fort.
	Zeitereignis [TimeEvent] (Aktionsknoten)	Des Zeitereignises wird auch als Signalereignisaktion definiert. Verwendet wird es aber nur, wenn ein zeitliches Ereignis eintritt.
	Verzweigung [DecisionNode] (Kontrollknoten)	Die Verzweigung stellt eine exklusive Entscheidung dar, bei der nur ein nachfolgender Pfad ausgeführt wird.

Grafische Darstellung	Element	Beschreibung
	Zusammenführung [MergeNode] (Kontrollknoten)	Die Zusammenführung ist ein exklusives Oder. Mit diesem Knoten werden einzelne Pfade zu einem weiterführenden Prozesspfad konsolidiert.
	Gabelung [ForkNode] (Kontrollknoten)	Eine Gabelung ist ein logisches Und und sendet für jede eingehende Marke an jedem Ausgang eine Marke weiter.
	Vereinigung [JoinNode] (Kontrollknoten)	Eine Vereinigung synchronisiert die eingehenden Prozesspfade. Erst wenn eine Marke in jedem Eingang vorhanden ist, wird eine Marke für die weitere Verarbeitung weitergegeben.
	Objektknoten [ObjectNode] (Objektknoten)	Mit einem Objektknoten wird ein Objekt von einer Aktion an die nachfolgende übermittelt. Das Element kann aber auch als Eingabe und Ausgabe der Aktivität verwendet werden.
	Objektknoten (Pin) [ObjectNodePin] (Objektknoten)	Anstelle eines separaten Elements kann ein Objektknoten auch direkt an eine Aktion geheftet werden, womit ein Objektfluss modelliert wird.
	Datenspeicher [DataStore] (Objektknoten)	Ein Datenspeicher stellt im Gegensatz zu einem Objekt einen dauerhaften Speicher für Informationen dar.
	Parametergruppe [ParameterSet] (Objektknoten)	Eine Parametergruppe kapselt ein oder mehrere Objektknoten. Mehrere Parametergruppen stellen Alternativen im Sinne eines exklusiven Oders dar.
	Aufrufaktion [CallAction] (Aktionsknoten)	Eine Aufrufaktion stellt eine Verknüpfung zu einer anderen Aktivität, mit einem eigenen Ablauf, dar.
	Verantwortungsbereich [ActivityPartition]	Der Verantwortungsbereich gibt an, wer oder was für die Ausführung einer Aktion zuständig ist. Alternativ kann dies auch mit runden Klammern „(Partition)“ in einer Aktivität angegeben werden.

Grafische Darstellung	Element	Beschreibung
	Bereich [ExpansionRegion]	Mit einem Bereich wird ein Teil des Ablaufs umschlossen und mit weiteren Verhaltensweisen versehen werden, beispielsweise eine parallele oder iterative Verarbeitung oder ein Abbruch einer Bearbeitung mittels einer Signalereignisaktion.
	Vor und Nachbedinungen [pre- and postconditions]	Vor- und Nachbedingungen geben zusätzlichen Faktoren an, damit eine Aktion ausgeführt wird: <<localPrecondition>> bei einer Vorbedingung und bei einer Nachbedingung <<localPostcondition>>.
	Ausgehender Konnektor [Outgoing Connector] (Konnektorknoten)	Der ausgehende Konnektor stellt einen Sprungpunkt dar, der auf einen gleichnamigen eingehenden Konnektor verweist.
	Eingehender Konnentor [Incoming Connector] (Konnektorknoten)	Der eingehende Konnektor stellt einen Zielpunkt für einen ausgehenden Konnektor dar. Beide Konnektoren zusammen werden auch als normaler Kontrollfluss aufgefasst (Oesterle 2005, S. 306).
	Kontrollfluss [ControlFlow]	Der Kontrollfluss gibt die Abarbeitungsreihenfolge in der Aktivität an.
	Objektfluss [ObjectFlow]	Der Objektfluss ist an einem Ende mit einem Objektknoten verbunden und zeigt dessen Weiterverarbeitung an.
	Ausnahmebehandlung [ExceptionHandler]	Eine Ausnahmebehandlung kann zusätzlich zum Kontroll- und Objektfluss angegeben werden. Dieser Pfad wird verfolgt, wenn ein Fehler in der Ausführung der Aktion vorlag.
	Assoziation	Eine Assoziation verknüpft Vor- und Nachbedingungen mit einer Aktion.

Die englischsprachigen Bezeichnungen, in den eckigen Klammern, stammen aus den Spezifikationen der OMG (2010, S. 430-434) und die Übersetzung ins Deutsche basiert auf Staud (2010, S. 215-217). Mit den Angaben in runden Klammern werden die Elemente in Gruppen

zusammengefasst, um die Verknüpfungsmöglichkeiten darzustellen. Die Elemente in Tabelle 5 widersprechen bei einem Element den Spezifikationen. Das Zeitereignis wird als eigenständiges Objekt aufgefasst, auch wenn dieses als Signalereignisaktion definiert ist (OMG 2010, S. 430). Die Signalereignisaktion erwartet aber ein externes Ereignis, während das Zeitereignis eine Zeitspanne abbildet.

Tabelle 6: Verknüpfungsmöglichkeiten im UML Aktivitätsdiagramm

Nach Von	Start knoten	Konnektor-knoten	Aktions-knoten	Objekt-knoten	Kontrol lknoten	End knoten
Start knoten		→	→	→	→	→
Konnektor-knoten		→	→	→	→	→
Aktions-knoten		→	→	→	→	→
Objekt-knoten		→	→	→	→	→
Kontroll-knoten		→	→	→	→	→
End knoten						

Aus den Verknüpfungsmöglichkeiten in Tabelle 6 kann eine Matrix abgeleitet werden; analog zu Abbildung 2 bei der eEPK. Im Gegensatz zur eEPK müssen aber nicht die konkreten Vorgänger und Nachfolger betrachtet werden, sondern die Anzahl. Werden Transformationsregeln aufgestellt, müssen diese Gestaltungsmöglichkeiten berücksichtigt werden.

Tabelle 7: Anzahl möglicher Vorgänger und Nachfolger der Elemente des UML Aktivitätsdiagramms

Anzahl Vorgänger	**0**	**0**	**0**	**1**	**1**	**1**	**n**	**n**	**n**
Anzahl Nachfolger	**0**	**1**	**n**	**0**	**1**	**n**	**0**	**1**	**n**
Elemente									
Startknoten		→	→						
Flussende				→			→		
Aktivitätsende				→			→		
Aktion		→	→	→	→	→	→	→	→
Signalereignisaktion		→	→	→	→	→	→	→	→
Zeitereignis		→	→	→	→	→	→	→	→
Signalsendeaktion		→	→	→	→	→	→	→	→
Verzweigung						→			→
Zusammenführung								→	→
Gabelung						→			→
Vereinigung								→	→
Objektknoten		→	→	→	→	→	→	→	→
Objektknoten (Pin)		→	→	→			→		
Datenspeicher		→	→	→	→	→	→	→	→
Aufrufaktion		→	→	→	→	→	→	→	→
Ausgehender Konnektor				→					
Eingehender Konnentor		→							

Bei der Zusammenführung (OMG 2010, S. 399) und Vereinigung (OMG 2010, S. 394) ist nach den Spezifikationen nur ein Nachfolger vorgesehen. Die Verzweigung und Zusammenführung hat aber das gleiche grafische Symbol. Hat dieses Symbol mehrere Vorgänger und Nachfolger erfüllt es beide Funktionen. Entsprechend ist in Tabelle 7 für beide Elemente eine $n:n$ Beziehung möglich.

Der „Objektknoten (Pin)“ ist ein Element wird direkt an eine Aktion angeheftet. Daher besitzt dieses entweder nur Eingänge oder nur Ausgänge. Wenn eine Aktivität mehrere Objektknoten als Pin hat, dann müssen alle eine Marke erhalten, damit die Aktion ausgeführt wird. Entsprechend wird eine Marke an jeden Objektknoten gesendet, die an den Ausgang der Aktion angeheftet sind. Die Objektknoten können auch mittels den Parametergruppen gruppiert werden. Alle Objektknoten einer Parametergruppe benötigen dann ebenfalls eine Marke (UND Bedingung), um die Aktion auszuführen. Mehrere Parametergruppen schließen sich jedoch gegenseitig aus (XOR Bedingung). Für die Ausführung der Aktivität benötigt müssen nur die Objektknoten von einer Parametergruppe eine Marke erhalten. Ausgangsseitig werden ebenfalls nur die Objektknoten einer Parametergruppe mit Marken versorgt.

Tabelle 7 und dessen Erläuterungen stellen die Grundlage für die Herleitung der Transformationsregeln dar. Ebenfalls kann die Tabelle für die syntaktische Prüfung der Transformationsregeln dienen. Für eine semantische Prüfung muss die Bedeutung eines Elements im UML Ablaufdiagramm berücksichtigt werden, wodurch Transformationsregeln notwendig sind, die sich nicht aus dem Schema der Tabelle ableiten.

Gesondert muss noch der Bereich betrachtet werden, der für eine parallele oder iterative Verarbeitung genutzt werden kann. Liegt diese Verarbeitung vor, werden am Rand des Bereichs vier Pins angebracht. Enthält ein Bereich nur eine Aktion, ist eine verkürzte Darstellung möglich (Oestereich 2006, S. 316); die vier Pins werden am Rand der Aktion platziert (OMG 2010, S. 384).

Die Ausnahmebehandlung kann in einem Bereich oder direkt bei einer Aktion verwendet werden. Tritt ein Ereignis ein, wird die Ausführung einer Aktionen (OMG 2010, S. 375) beendet oder aller Aktionen innerhalb des Bereiches (OMG 2010, S. 393).

2.4 Über die Simulierbarkeit von Prozessmodellen

Bei der Simulation von Prozessmodellen stellt sich die Frage: Ist das Modell simulationswürdig? Hierfür führt Bangsow (2008, S. 11) Gesichtspunkte auf, um dies einzuschätzen:

- „Fehlen analytischer mathematischer Modelle (viele Variablen)
- hohe Komplexität, viele zu berücksichtigende Einflüsse
- Datenunsicherheit
- schrittweises Ausloten von Systemgrenzen
- wiederholtes Verwenden des Simulationsmodells“

Als semiformale Modelle verfügen Prozessmodelle nicht über ein analytisches mathematisches Modell. Eine hohe Komplexität ergibt sich, wenn mehr als ein Prozessmodell simuliert werden soll oder das Prozessmodell mehrere Verzweigungen und Variablen aufweist.

Datenunsicherheit betrifft das Prozessmodell und das daraus erzeugte Simulationsmodell. Dies tritt auf die Prozesse zu, die vornehmlich von Menschen ausgeführt werden. Bei diesen kann nicht von einer festen Bearbeitungszeit für die Aktivitäten ausgegangen werden.

Der Gesichtspunkt zur Auslotung von Systemgrenzen ist abhängig von den Zielen der Simulation. Dies kann erfolgen, wenn einzelne Parameter für die Simulation geändert werden, um die Auswirkungen zu analysieren. Ob ein Simulationsmodell wieder verwendet wird, ist abhängig vom Modellierer.

Aus dieser Analogie lässt sich schlussfolgern, dass Prozessmodelle als simulationswürdig aufgefasst werden können. Das größte Problem, um Erkenntnisse aus der Simulation auf die Realität zu übertragen, liegt jedoch in der Durchführung der Prozessmodelle in der Realität. Wird die tägliche Arbeit nicht nach den Prozessmodellen durchgeführt, dann können die Erkenntnisse nicht in die Realität übertragen werden.

Wie auch Neumann et al. (2005, S. 436) aufführen, ist nicht jeder Prozess für eine Simulation geeignet. Hierfür führen die Autoren vier Anforderungen an simulationsfähige Prozesse auf:

- Der Prozess muss über längere Zeit stabil sein.
- Der Prozess sollte in einer Periode hinreichend ausgeführt werden.
- Für die Simulation von Prozessen werden mehr Daten benötigt als für reine Prozessmodelle. Diese notwendigen Daten sollten effizient erhoben werden können. Ist eine manuelle Datenerhebung notwendig, so kann dies einen erheblichen Aufwand bedeuten.
- Die Simulation muss entsprechende Ziele haben, um den Aufwand und die Kosten für die Simulation zu rechtfertigen.

Mit dem ersten Punkt wird die Thematik angesprochen, wie lange ein Prozess gültig ist, bis dieser durch einen neueren ersetzt wird. Da eine Simulation über einen längeren Zeitraum durchgeführt wird, aber auch die Durchführung einer Simulationsstudie Zeit in Anspruch nimmt, sollte der Prozess für mindestens diese Zeitspanne gültig sein. Der zweite Punkt wurde bereits bei den obigen Ausführungen unter dem Gesichtspunkt der Datenunsicherheit aufgegriffen. Zusätzlich zu diesem sollten nur Prozesse simuliert werden, die öfters ausgeführt werden. Wird ein Prozess beispielsweise in einem Monat nur drei Mal ausgeführt, so ist dieser Prozess weniger für eine Simulation geeignet,

als ein Prozess der über hundert mal im Monat ausgeführt wird. Der dritte Punkt spricht die Datenerhebung für die Simulation an und dass die notwendigen Daten für eine Simulation in der Regel nicht in Prozessmodellen vorzufinden sind. Die Datenerhebung kann selbst einen erhebliche Aufwand bedeuten, der mit dem vierten Punkt zusammenhängt. Aufgrund des zeitlichen Aufwandes und den Kosten für die Datenerhebung und der Vorbereitung des Simulationsmodells muss ein entsprechender Nutzen dem gegenüberstehen. Ist der Aufwand für die Vorbereitung und Erstellung des Simulationsmodells zu hoch, so eignen sich Prozesse nicht für eine Simulation aus betriebswirtschaftlicher Sicht.

Auch wenn Prozessmodelle prinzipiell für die Simulation geeignet sind, ist der entscheidende Faktor für die Durchführung einer Simulation der betriebswirtschaftliche Nutzen. Wenn der Aufwand für die Durchführung einer Simulation reduziert werden kann, dann können mehr Prozesse für eine Simulation in Betracht gezogen werden.

Ein Problem an den bisherigen Ausführungen ist, dass es sich um qualitative Aussagen handelt. In der Literatur, die sich auf die Simulation von Geschäftsprozessen bezieht, fehlen quantitative Aussagen, wann ein Geschäftsprozess simulationswürdig ist. Für eine quantitative Bestimmung, ob ein Geschäftsprozess simulationswürdig ist, wird ein Ansatz vorgeschlagen, der auf die Komplexität von Geschäftsprozessmodellen zurückgreift. Hierfür wird auf Metriken zurückgegriffen, die in Mendling (2008) zur Vorhersage von Fehlern in Prozessmodellen entwickelt wurde. Hierbei wird die These aufgestellt, dass komplexe Prozessmodelle eher Fehler beinhalten als einfache Prozessmodelle. Je höher die Wahrscheinlichkeit für das Auftreten von Fehlern in einem Prozessmodell ist, desto komplexer ist ein Prozessmodell. Ein komplexes Prozessmodell erfüllt eher das Kriterium der Simulationswürdigkeit als ein einfaches. Für die Messung der Komplexität sollen nahezu die gleichen Kriterien herangezogen werden, wie sie bei Mendling (2008, S. 117-133) definiert sind.

Mendling (2008, S. 131) definiert 28 Metriken, um die Fehleranfälligkeit zu messen. Diese sind jeweils mit einer Hypothese verbunden und leiten sich aus verschiedenen Literaturquellen ab. Ein Beispiel für Hypothese der Metrik „density" ist: „An increase in $\Delta(G)$ should imply an increase in error probability of the overall model." (Mendling 2008, S. 120). Die Metriken wurden anhand der dafür definierten Hypothese überprüft. Hierbei stellte sich heraus, dass 2 Hypothesen nicht bestätigt werden konnten, beziehungsweise falsifiziert wurden. Die 26 nicht falsifizieren Metriken sind in Tabelle 8 aufgeführt.

Tabelle 8: Metriken zur Messung der Fehleranfälligkeit von EPK Modellen

Metrik	Metrik
Size S_N	Size S_A
Size S_E	Size *diam*
Size S_{E_S}	density Δ
Size $S_{E_{Int}}$	density *CNC*
Size S_{E_E}	max. connector degree $\widehat{d_C}$
Size S_F	separability Π
Size S_C	sequentiality Ξ
Size $S_{S_{XOR}}$	structuredness Φ
Size $S_{J_{XOR}}$	mismatch *MM*
Size $S_{S_{AND}}$	heterogeneity *CH*
Size $S_{J_{AND}}$	control flow complexity *CFC*
Size $S_{S_{OR}}$	cyclicity *CYC*
Size $S_{J_{OR}}$	token splits *TS*

Die einzelnen Kriterien werden in sechs Kategorien unterteilt: „size", „density", „partitionability", connector „interplay", „cyclicity" und „concurrency". Mehr als die Hälfte der Metriken sind der ersten Kategorie der Größe („size") zuzuordnen. Hierbei ist zu beachten, dass sich die Metriken, insbesondere dieser Kategorie, auf EPK Modelle

beziehen. Für andere Notationen muss die Eignung dieser („size“) Metriken erneut überprüft werden und gegebenenfalls angepasst werden.

Hierbei ist auch zu beachten, dass ein direkter Vergleich der Anzahl der Elemente eines Geschäftsprozesses, der in zwei verschiedenen Notationen modelliert wurde, nicht direkt erfolgen kann. So werden bei einem EPK Modell Ereignisse nach den Funktionen definiert (Staud 2010, S. 205), während bei einem UML Diagramm Objekte im Prozessablauf vorzufinden sind (Staud 2010, S. 204). Der gleiche Geschäftsprozess kann in verschiedenen Notationen unterschiedlich komplex erscheinen, weshalb die Werte der Metriken nicht direkt miteinander verglichen werden sollten.

Die einzelnen Metriken von Mendling werden an dieser Stelle nicht erläutert, da nur ein Vorschlag für deren Anwendung zur Bestimmung der Simulationswürdigkeit vollzogen wird. Für eine Vertiefung wird auf Mendling (2008) verwiesen. Neben diesen Metriken ist, wie bereits aufgeführt, die Häufigkeit der Ausführung eines Geschäftsprozesses ein wesentliches Merkmal, um eine Aussage zu treffen, wann ein Geschäftsprozess simulationswürdig ist.

Zur Bestimmung der Simulationswürdigkeit, ist es notwendig, konkrete Angaben für die Metriken der einzelnen Geschäftsprozess-modellierungsnotationen festzulegen. Hierfür wird die Durchführung einer Befragung vorgeschlagen, bei der erfahrene Personen einbezogen werden, die sich mit der Simulation von Geschäftsprozessen beschäftigen. Als Ergebnis wäre eine Aussage darüber zu erwarten, welchen Wert eine Metrik annehmen muss, damit nach dieser das Geschäftsprozessmodell als simulationswürdig eingeschätzt wird. Aus den Ergebnissen der Aussagen kann daraufhin ein gemeinsamer Konsens gebildet werden. Für die eigentliche Bestimmung wird vorgeschlagen, dass nicht alle Metriken einen gewissen Schwellwert überschreiben müssen, sondern dass n aus m Metriken die Schwelle

erreichen müssen. Alternativ ist zu prüfen, ob Metriken unterschiedlich zu gewichten wären.

Eine derartige Studie ist aber nicht Gegenstand dieser Arbeit und soll lediglich als Ansatzpunkt und offene Forschungslücke aufgefasst werden, um die Simulationswürdigkeit von Geschäftsprozessmodellen quantitativ bestimmen zu können.

Zur Aufdeckung der Forschungslücke und Arbeiten zur Simulation von Geschäftsprozessen

Wie eine Vielzahl an Arbeiten zeigt, sind Geschäftsprozesse Gegenstand von Simulationsstudien. Diese müssen daher von den Autoren als simulationswürdig aufgefasst worden sein. Eine Review über diese Arbeiten soll aufzeigen, welche Aspekte für das Thema der Arbeit abgeleitet werden können, beziehungsweise den Forschungsbedarf aufzeigen. Hierfür wurde eine Literaturrecherche durchgeführt. In Anlehnung an Gräning et al. (2011, S. 226-227) erfolgt, ebenfalls mit Bezug auf Fettke (2006), eine Offenlegung der durchgeführten Literaturrecherche.

Für das Auffinden von relevanter Literatur wurden verschiedene Datenbanken, mit entsprechenden Suchbegriffen, durchsucht. Die verwendeten Suchbegriffe waren jeweils in Kombination mit dem Wort „Simulation“: „Prozess“, „Geschäftsprozess“, „Process“ und „Business Process“. Neben dieser direkten Suche wurden die Literaturverzeichnisse der gefundenen Artikel gesichtet, um weitere Arbeiten aufzufinden.

Die primär verwendeten Datenbanken der Recherche waren:

- Ebsco Host - Business Source Premier
- CiteSeerX
- Science Direct
- Springer Link
- Winter Simulation Conference Archive
- WISO Datenbank

Ebenfalls in die Suche eingeschlossen waren die zwei wichtigsten Zeitschriften der Wirtschaftsinformatik (Fettke, 2011, S. 260), WIRTSCHAFTSINFORMATIK und HMD. Berücksichtigt werden auch Ausführungen der Einführungsliteratur aus dem Gebiet des Geschäftsprozessmanagements. Die Einordnung der durchgeführten Literaturrecherche, in das Schema zur Charakterisierung für Reviews von Fettke (2011, S. 259), ist in Tabelle 9 aufgeführt. Die Einordnung zu den Charakteristika sind jeweils markiert.

Das Ergebnis der Literaturrecherche wird nachfolgend natürlich-sprachlich aufgeführt, ohne die Verwendung von statistischen Verfahren. Bei dieser Betrachtung werden Arbeiten herangezogen, die über Erfahrungen mit ausgeführte Simulationsstudien berichten. Ziel der Auswertung ist das Aufführen von Mustern, die bei den einzelnen Arbeiten auftreten. Nach Fettke (2011, S. 259) ist die Literaturrecherche aus inhaltlicher Sicht den „zentralen Themen“ zuzuordnen.

Tabelle 9: Charakterisierung der Literaturrecherche

Charakteristik		Kategorie			
Typ		natürlichsprachlich		mathematisch-statistisch	
Fokus		Forschungs-ergebnis	Forschungs-methode	Theorie	Erfahrung
Ziel	**Formulierung**	nicht expliziert		expliziert	
	Inhalt	Integration	Kritik	zentrale Themen	
Perspektive		neutral		Position	
Literatur	**Auswahl**	nicht expliziert		expliziert	
	Umfang	Schlüsselarbeiten	repräsentativ	selektiv	methodisch
Struktur		historisch	thematisch	methodisch	
Zielgruppe		Allgemeine Öffentlichkeit	Praktiker	Forscher im Allgemeinen	Spezialisierte Forscher
Zukünftige Forschung		nicht expliziert		expliziert	

Nach Fettke (2011, S. 259)

Die Perspektive bezieht sich auf die Sicht aus einer bestimmten Forschungsrichtung oder aus Sicht eines Paradigmas. Hier erhebt die Recherche den Anspruch neutral die Arbeiten zu betrachten. Dies trifft auch auf alle untersuchten Reviews von Fettke (2011, S. 262) zu, da sich in der Wirtschaftsinformatik noch keine Paradigmata durchgesetzt haben (Fettke 2011, S. 264). Die Einordnung der Arbeit in die gestaltungsorientierte Wirtschaftsinformatik hat keine Auswirkungen auf die durchgeführte Literaturrecherche.

Die Auswahl der Literatur wurde bereits, orientiert an Gräning et al. (2011, S. 226-227) aufgeführt. Durch die Wahl bestimmter Suchbegriffe ist der Umfang daher als selektiv einzuordnen. Die Aufführung der gefundenen Literatur erfolgt thematisch mit der Zielgruppe von spezialisierten Forschern. Die Aufdeckung von Gemeinsamkeiten soll

der Identifikation der Forschungslücke dienen, welche durch die vorliegende Arbeit geschlossen werden soll, womit eine Nennung der zukünftigen Forschung erfolgt.

Die nachfolgend aufgeführten Arbeiten können in zwei Kategorien unterteilt werden. Zum einen kann ein Prozessmodell für die Simulation direkt verwendet werden, zum anderen kann dieses für die Erstellung eines Simulationsmodells als Grundlage dienen.

Beispiele für eine direkte Simulation eines Geschäftsprozessmodells sind bei Neumann et al. (2005), Allweyer (2005, S. 243-261) und Gadastch (2010, S. 216-248) vorzufinden. Diese Ansätze basieren auf der von Neumann et al. (2005, S. 441) aufgeführten Attributierung der Elemente von Geschäftsprozessmodellen. Hierbei werden simulationsrelevante Attribute den Elementen des Geschäftsprozessmodells hinterlegt. Während Allweyer (2005, S. 248) die Software ARIS für die Simulation einer eEPK verwendet, wird bei Gadatsch (2010, S. 235) die Software Process Charter für die Simulation eines Workflow-Diagramms verwendet. Es erfolgt jeweils eine Hinterlegung mit simulationsrelevanten Attributen.

Das Problem, dass an dieser Stelle auftritt, ist, dass nur die Elemente mit simulationsrelevanten Attributen hinterlegt werden können, die bereits im Geschäftsprozessmodell vorhanden sind. Während Ressourcen zwar auf einfache Weise hinzugefügt werden können, ist die Reservierung von Ressourcen mit einem bestehenden Geschäftsprozessmodell nicht möglich. Hierfür wird auf das Zahnarzt-Gedankenexperiment in Kloos et al. (2009, S. 93) verwiesen (Abbildung 9). Das Gedankenexperiment besteht aus drei sequenziellen Funktionen, bei denen unterschiedliche Personen für die Ausführung notwendig sind (Abbildung 9a). Alle drei Funktionen werden jeweils im gleichen Raum durchgeführt, der im dargestellten Beispiel hinzugefügt werden muss (Abbildung 9b). Das Gedankenexperiment in Abbildung 9 wird als eEPK dargestellt, wobei das Symbol für den Behandlungsraum, eine stationäre Ressource repräsentieren soll.

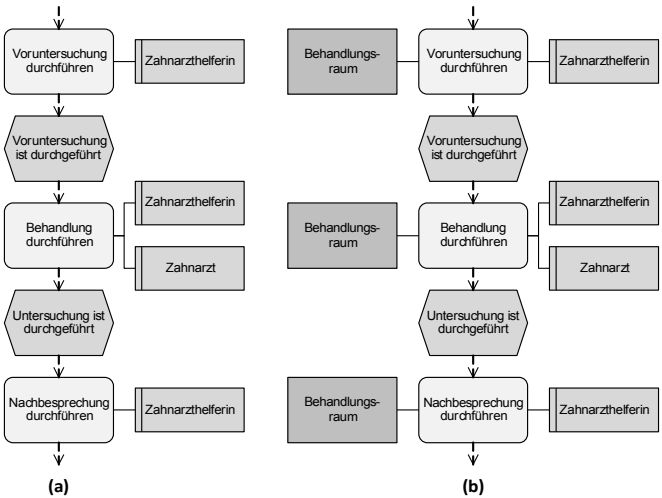


Abbildung 9: Zahnarzt Gedankenexperiment

Als eine Annahme wird dem Gedankenexperiment unterstellt, dass die beanspruchten Ressourcen mehr als einmal zur Verfügung stehen. Wird diese sequenzielle Abfolge simuliert, dann ist nicht sichergestellt, dass alle drei Funktionen für jeden Patienten in einem Block durchgeführt werden, da keine Reservierung beziehungsweise Steuerung der Ressourcen erfolgt. So besagt das Gedankenexperiment, dass Patient A den Behandlungsraum betritt, und die Voruntersuchung erhält. Anschließend muss der Patient den Raum wieder verlassen, damit Patient B die Voruntersuchung erhalten kann. Patient B verlässt daraufhin den Behandlungsraum und Patient A erhält die Behandlung. Dieses Gedankenexperiment steht aber im Gegensatz zur Realität, bei der Patient A erst nach Erhalt der Nachbesprechung den Behandlungsraum verlässt und danach Patient B die Behandlung erhält.

Geschäftsprozessmodelle beinhalteten für diese Steuerung der Ressourcen keine Elemente und Mechanismen, da diese nicht im Fokus der Geschäftsprozessmodellierung liegen (Gadatsch 2010, S. 112). Aufgrund der fehlenden Steuerung der Ressourcen sind Geschäftsprozessmodelle nicht geeignet, direkt simuliert zu werden. Diese Aussage muss noch relativiert werden. Unter der Annahme, dass

Funktionen nicht am Block durchgeführt werden müssen, ist die direkte Simulation möglich. Sobald jedoch Räumlichkeiten berücksichtigt werden, ist besonders zu prüfen, ob das Verhalten des Simulationsmodells der Realität entspricht.

Die zweite Kategorie betrachtet Prozessmodelle als Grundlage, um daraus ein Simulationsmodell abzuleiten. Die Ergebnisse der Literaturrecherche zu dieser Kategorie werden nachfolgend aufgeführt.

Greasley (2003) verwendet die Geschäftsprozesssimulation im Rahmen eines Business Process Reengineering Ansatzes für den Prozess der Aufsicht von Sträflingen. Hierfür wird als konzeptionelles Modell eine Art Ablaufdiagramm („process map“) verwendet. Dieses bildet die Grundlage, um die notwendigen quantitativen Daten zu ermitteln, die für das Simulationsmodell benötigt werden. Das Ablaufdiagramm und die erhobenen quantitativen Daten werden anschließend in ein Simulationsmodell überführt, um die eigentliche Simulationsstudie durchzuführen. Als Zielumgebung gibt Greasley (2003, S. 413) die Software Arena an.

Einen ähnlichen Simulationsansatz, für die Notaufnahme in einem Krankenhaus, verwenden Seyfert und Kavermann (2006). Grundlage für das Simulationsmodell stellt eine Behandlungsmatrix dar, welche die einzelnen Behandlungsschritte für Krankheiten beinhaltet. Die Reihenfolge der Behandlung ist nicht direkt im Simulationsmodell enthalten, sondern jede Marke enthält den individuellen Verlauf durch das Simulationsmodell. Das konstruierte Modell hat nach Seyfert und Kavermann (2006, S. 42) die Einschränkung, dass dieses nur mit Prozessen verwendbar ist, die in Form der Behandlungsmatrix abbildbar sind. Als Simulationsumgebung geben Seyfert und Kavermann (2006, S. 40) die heutige Software Plant Simulation an.

In einem anderen Artikel beschreibt Greasley (2000) ebenfalls die Simulation von zwei Prozessen aus dem polizeilichen Umfeld. Im Fokus des Artikels stehen Kosten, die mittels der Simulation ermittelt

werden können. Hinsichtlich der Erzeugung des Simulationsmodells wird lediglich aufgeführt, dass Prozesse modelliert wurden und diese als konzeptuelles Modell für das Simulationsmodell verwendet werden (Greasley 2000, S. 2005).

Damij und Damij (2008) untersuchen in ihrem Artikel die Möglichkeit Aktivitätstabellen („activity table“) mit Simulation zu kombinieren, um Geschäftsprozesse zu verbessern. Hierfür untersuchen sie einen Operationsprozess in einem Krankenhaus. Die Aktivitätstabelle ist dadurch gekennzeichnet, dass in den Zeilen die durchzuführenden Funktionen und den Spalten die Personen gruppiert nach Abteilungen abgetragen werden (Damij und Damij 2008, S. 307). In die einzelnen Zellen wird der Ablauf des Prozesses eingetragen. Die Aktivitätstabelle kann mit einem Swimlane Diagramm (Gadatsch 2010, S. 85) verglichen werden. Nach Damij und Damij (2008, S. 308) muss die Aktivitätstabelle in ein Ablaufdiagramm („flow chart“) überführt werden, damit aus diesem eine Simulationsumgebung entwickelt werden kann. Hinsichtlich des Informationsgehalts sind die Aktivitätstabelle und das Ablaufdiagramm jedoch identisch. Das Ablaufdiagramm soll anschließend mit simulationsrelevanten Daten gefüllt werden. Wie dies jedoch im Rahmen des Ablaufdiagramm erfolgt, wird von den Autoren nicht angegeben. Damij und Damij (2008, S. 308) geben aber als Simulationssoftware iGrafx an.

Dickmann et al. (2005) beschreiben einen Ansatz, mit dem quantitative Daten aus implizitem Wissen über einen Prozess gewonnen werden können. Im Blickpunkt haben die Autoren hierbei nicht Geschäftsprozesse an sich, sondern den Softwareentwicklungsprozess, der simuliert werden soll. Hierfür schreiben Dickmann et al. (2005, S. 276), dass das Simulationsmodell aus existierenden konzeptuellen Prozessmodellen abgeleitet wird, während quantitative Parameter aus einer Expertenbefragung gewonnen werden. Wie die Prozessmodelle abgeleitet werden und welche Simulationsumgebung zum Einsatz kommt, wird von den Autoren aber nicht aufgeführt.

Desel und Erwin (2000) verwenden bei ihrem Aufsatz „Modeling, Simulation and Analysis of Business Processes“ Petri-Netze zur Modellierung von Geschäftsprozessen, die sie für die Simulation heranziehen. Hierfür verwenden die Autoren ein drei-stufiges Verfahren. Beim ersten Schritt erfolgt die reine Modellierung des Geschäftsprozesses mittels eines Petri-Netzes. Im zweiten Schritt werden aus dem Petri-Netz alle relevanten Prozessabläufe („runs“) abgeleitet. Zeiten und Kosten werden im dritten Schritt den relevanten Prozessabläufen zugewiesen. Die Analysen zu Kosten und Zeit erfolgen daraufhin an diesen Prozessabläufen. Trotz des Titels, der eine Simulation von Geschäftsprozessen andeutet, entspricht dieses Verfahren der Analyse nicht der in dieser Arbeit aufgeführten Definition einer Simulation.

Aguilar et al. (1999) führten die Simulation der Geschäftsprozesse einer Bank durch. Aus dem Artikel geht hervor, dass die Geschäftsprozesse der Bank modelliert wurden und zur Analyse der Geschäftsprozesse Simulation eingesetzt wurde (Aguilar 1999, S. 1386). Ein Beispiel für ein Simulationsmodell wird von den Autoren zum Prozess „Deliver Credits“ dargestellt. Aus den Ausführungen lässt sich schließen, dass die modellierten Geschäftsprozesse als Grundlage für die Erstellung des Simulationsmodells herangezogen wurden. In welcher Notation die Geschäftsprozesse modelliert wurden und wie die Überführung in die Simulationsumgebung Arena (Aguilar et al. 1999, S. 1388) vollzogen wurde, geben die Autoren jedoch nicht an.

Die vorgestellten Ansätze sind geprägt von zwei Hauptproblemen. Zum einen weisen Prozessmodelle einen höheren Detaillierungsgrad hinsichtlich der Anzahl der Prozessschritte auf, zum anderen benötigen Simulationsmodelle detailliertere und quantifizierte Daten für die Prozessschritte (Kloos et al. 2009; Kloos et al. 2010). Ebenfalls muss, wenn ein modellierter Geschäftsprozess verwendet wird, eine Überführung in eine Simulationsumgebung erfolgen und simulatonsrelevante Daten müssen über verschiedene Methoden erhoben werden. Ein allgemeiner Ansatz für diese beiden Punkte liegt nicht vor. Somit

stellt jeder Modellierer vor dem Problem Transformationsregeln für die modellierten Geschäftsprozesse aufzustellen, um eine Abbildung in einer Simulationsumgebung zu ermöglichen.

Nachfolgend werden noch weitere Arbeiten betrachtet, die sich mit der Simulation von Geschäftsprozessen befassen, jedoch kein Geschäftsprozessmodell als Grundlage verwenden.

An und Jeng (2005) legen ein Ablaufdiagramm („flow chart“) eines Geschäftsprozesses zugrunde, um dieses für ein diskretes ereignisorientiertes Simulationsmodell und darauf aufbauendes System Dynamics Modell zu verwenden. Das Simulationsmodell wird mit simulationsrelevanten Daten angereichert. Für das System Dynamics Modell wird das Simulationsmodell herangezogen. Ziel der Simulation mit dem System Dynamics Modell ist das Nachverfolgen von Änderungen der Metriken des Geschäftsprozessmodells. Wie An und Jeng (2005, S. 273) aufführen, wird das System Dynamic Modell nicht automatisch erzeugt, da betriebswirtschaftliche Kenntnisse über Kausalzusammenhänge mathematisch zu formulieren sind.

Heavy und Ryan (2006, S. 803) schlussfolgern mit Bezug auf Ryan und Heavy (2006), dass Prozessmodellierungswerkzeuge sowie Prozessmodellierungsnotationen nicht die konzeptuelle Modellierung für Simulationsprojekte unterstützen. Die Untersuchung schloss unter anderem das UML Aktivitätsdiagramm und die eEPK mit ein (Heavy und Ryan 2006, S. 802-803). Als Schlussfolgerung dessen, konzipierten die Autoren die Methode „Simulation Activity Diagram“. Diese Methode ist jedoch dafür gedacht, die konzeptuelle Modellierung für ein Simulationsprojekt zu unterstützen. Es ist mit dieser nicht vorgesehen, bestehende Geschäftsprozessmodelle als Grundlage zu verwenden.

Ein Ansatz, welcher in Richtung eines Transformationsmodells geht und verschiedene Simulationssysteme adressiert wird in Ehm et al. (2009), Schönherr und Rose (2009) sowie Schönherr und Rose (2010) aufgeführt. Der Ansatz beabsichtigt die Modellierung von Simulationsmodellen mittels SysML zur Abbildung von Produktionsprozessen. Der Schwerpunkt im Rahmen eines Vorgehensmodells zur Durchführung einer Simulationsstudie liegt bei diesem Ansatz bei der Systemdefinition, der Formulierung des konzeptuellen Modells und der Überführung in angemessene Simulationssprachen (Ehm et al. 2009, S. 1695). Bezogen auf die Simulationssprachen identifizieren Ehm et al. (2009, S. 1696), dass einige einem hohen Abstraktionsgrad und andere hingegen sehr detailliert ein Simulationsmodell beschreiben. Beide Abstraktionsgrade sollen hierbei vom SysML Ansatz unterstützt werden, soweit die Simulationssprache angemessen ist.

Der Ansatz sieht vor, dass ein konzeptuelles Simulationsmodell mit SysML modelliert wird (Schönherr und Rose (2009, S: 1716). Die Modellierung erfolgt auf drei Ebenen, der Prozessebene, die Verhaltensebene und die Ausführungsebene, mit jeweils steigender Granularität (Schönherr und Rose 2010, S. 458-459). Nach Abschluss der Modellierung wird das SysML in ein internes Modell überführt. Das interne Modell wird daraufhin mittels Übersetzungskomponente in das Format der einzelnen Simulationsumgebungen überführt (Schönherr und Rose 2010, S. 455). Als Zielumgebungen werden von Schönherr und Rose (2009, S. 1712) Simul8, AnyLogic und eM-Plant (Plant Simulation) angeben.

Im Sinne der Zielsetzung der vorliegenden Arbeit gibt es Parallelen mit dem SysML Ansatz. Der SysML Ansatz adressiert auch mehrere Simulationsumgebungen. Dieser setzt jedoch eine Modellierung des Simulationsmodells mit SysML voraus. Bestehende Geschäftsprozessmodelle werden nicht als Grundlage verwendet. Wie Schönherr und Rose (2010, S. 459) aufführen, ist der Ansatz auch nicht für die Simulation von Geschäftsprozessen entwickelt worden, sondern für Produktionsplanung und -steuerung. Im Ausblick führen die Autoren

auf, dass sie die Krankenhauslogistik und das Bauwesen als weitere Domänen betrachten wollen. Aus Ehm et al. (2009, S. 1695) geht jedoch hervor, dass die Beschaffung von simulationsrelevanten Daten nicht von diesem Ansatz unterstützt wird.

Andere Artikel beschreiben Aspekte der Simulation von Geschäftsprozessen, gehen aber nicht auf die Generierung von Simulationsmodellen ein. So geben Barber et al. (2003) einen Überblick über Ansätze die eine Geschäftsprozesssimulation im Produktionsumfeld durchgeführt haben. Paul et al. (1998) geben ebenfalls einen Überblick über Artikel zur Geschäftsprozesssimulation. Die Autoren kommen aber zum Schluss, dass es nur wenig Artikel zur Simulation von Geschäftsprozessen gibt (Paul et al. 1998, 311).

Ebenfalls stellen Paul et al. (1998, S. 315-318) ein Vorgehensmodell („Framework") für die Geschäftsprozesssimulation vor. Lam und Lau (2004) beschreiben die Simulation eines Call Centers. Wie die Autoren aufführen, wurden keine bestehenden Geschäftsprozesse als Grundlage verwendet, sondern das Simulationsmodell direkt erstellt (Lam und Lau 2004, S. 485). Fetter und Thompson (1965) beschreiben die Simulation innerhalb eines Krankenhauses, geben aber nur an, dass drei Simulationsmodelle entwickelt wurden, jedoch nicht mit welcher Methode.

2.5 Notationen von Simulationsmodellen und -systemen

Die betrachteten Simulationssysteme werden zunächst mittels Studien von Hlupic (2000) sowie Jansen-Vullers und Netjes (2006) eingegrenzt.

Jansen-Vullers und Netjes (2006) untersuchen sechs Werkzeuge für die Geschäftsprozesssimulation; jeweils zwei Werkzeuge für Geschäftsprozessmodellierung, Geschäftsprozessmanagement und reine Simulationswerkzeuge. Für den Vergleich werden die Kriterien in drei Gruppen eingeteilt, die Leistungsfähigkeit der Modellierung, die Leistungsfähigkeit der Simulation und die Leistungsfähigkeit der Ergebnisanalyse. Gemäß Jansen-Vullers und Netjes (2006, S. 89-92) werden die Kriterien für die Modellierung am besten durch Geschäftsprozessmodellierungswerkzeuge erfüllt. Hervorzuheben ist das Abschneiden von Arena; als reines Simulationswerkzeug wird der zweite Platz bei diesen Kriterien erreicht. Die Kriterien zur Leistungsfähigkeit der Simulation und Ergebnisanalyse werden hingegen von den Simulationswerkzeugen dominiert. Gemäß dieser Studie sind reine Simulationswerkzeuge am besten geeignet, Geschäftsprozesse zu simulieren. Wird die Leistungsfähigkeit der Simulationswerkzeuge aber nicht benötigt, kann die Wahl auch auf ein Geschäftsprozessmodellierungswerkzeug fallen.

Bezogen auf die einzelnen Werkzeuge, schlussfolgern Jansen-Vullers und Netjes (2006, S. 93), dass ARIS als Vertreter der Modellierungswerkzeuge, Arena und das CPN Tool, beide als Vertreter der reinen Simulationswerkzeuge geeignete Kandidaten für eine Geschäftsprozesssimulation sind. Als Kritikpunkt an ARIS führen die Autoren aber auf, dass der (X)OR Konnektoren bei der EPK nicht eindeutig definiert ist und zu unvorhergesehenem Verhalten bei der Simulation führen kann (Jansen-Vullers und Netjes 2006, S. 88). Gegen das CPN Tool bringen Jansen-Vullers und Netjes (2006, S. 93) die Modellierung der Prozesse mittels Petri-Netzen vor, die für Prozessverantwortliche schwer zu verstehen seien und somit die Validierung des Modells erschweren. Als Ergebnis der Studie tritt Arena als ein geeignetes Werkzeug für die

Simulation von Geschäftsprozessen hervor. In diesem können können Modelle zwar einfach abgebildet werden, jedoch sind einige Elemente zu verschachtelt oder anzupassen (Jansen-Vullers und Netjes 2006, S. 88, 93).

Diese Erkenntnisse zu Arena decken sich mit Ergebnissen der Studie von Hlupic (2000). Diese untersucht die Verwendung und Anforderungen von akademischen und industriellen Nutzern an Simulationswerkzeuge. Die Stichprobe für die Untersuchung wurde aus Mitgliedern der „Simulation Study Group" und der „Operational Research Society of Great Britain" gezogen (Hlupic 2000, S. 1677). Eine Allgemeingültigkeit der Ergebnisse kann entsprechend nicht unterstellt werden. Von den befragten akademischen Nutzern verwendet etwa die Hälfte maximal 2 verschiedene Simulationssysteme. Das mit 44,4% am häufigsten verwendete System bei dieser Nutzergruppe ist Simul8. Wird jedoch Arena und die dahinter liegenden Simulationssprache SIMAN (Profozich und Sturrock 2005, S. 515) zusammengefasst, so werden diese genauso häufig verwendet. Bei industriellen Nutzern gibt es keine dominierende Simulationssoftware. (Hlupic 2000, S. 1680).

Wie Hlupic (2000, S. 1681) aber aufführt, müssen die Ergebnisse mit Vorsicht betrachtet werden, da diese einer kleinen Stichprobe zugrunde liegen. Ebenfalls ist hinsichtlich der Benutzer zu beachten, dass die Studie bereits über 10 Jahre in der Vergangenheit liegt. Für Arena spricht aber ebenfalls die Analyse von Russel Barton; welche Simulationswerkzeuge auf der Winter Simulation Conference in den Jahren 1997 und 2002 verwendet wurden. In diesen Jahren stellte Arena eines der verwendeten Werkzeuge dar (Barton et al. 2003, S. 2048). Für Arena kann die Aussage von Rossetti (2010, S. 10) als wahr anerkannt werden; Arena hat eine starke akademische und industrielle Nutzerbasis, auch wenn keine aktuellen Nutzerstatistiken auf akademischer und industrieller Ebene vorliegen.

Während bei Arena die Prämisse einer hohen Nutzerbasis angenommen werden kann, ist dies bei der zweiten betrachteten Simulationssoftware nicht durch eine Studie empirisch belegbar. Bei der zweiten Simulationssoftware handelt es sich um AnyLogic. Anhand einer durchgeführten Literaturrecherche mit dem Schlüsselwort „AnyLogic“, in den unter 2.4 aufgeführten Datenbanken, werden aber mehrere Arbeiten gefunden, die AnyLogic adressieren. Beispiele aus der Domäne der Logistik sind die Arbeiten von Almeder und Preusser (2007), Sharawi et al. (2006), Lloret et al. (2009) sowie Li und Li (2010). Aus der Domäne der Geschäftsprozesse kann auf die Artikel von Ardagna et al. (2008) sowie Schmietendorf und End (2009) verwiesen werden. Weitere Arbeiten, die AnyLogic als Simulationsumgebung verwenden sind Ehm et al. (2009), Kondratyev und Garifullin (2009) sowie Landau et al. (2004) und Borshchev et al. (2001). Ein Großteil der Arbeiten stammt aus dem Jahr 2006 und später. Aus diesen Arbeiten wird die These aufgestellt, der Einsatz von AnyLogic hat sich in den letzten Jahren verbreitet. Die aufgeführten Studien erstrecken sich nur bis ins Jahr 2006 und konnten die Verbreitung nicht beinhalten. Für die Simulation von Geschäftsprozessen mittels AnyLogic sprechen insbesondere die Artikel von Ardagna et al. (2008) sowie Schmietendorf und End (2009).

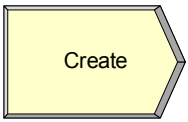
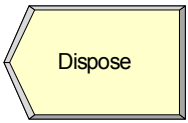
Für diese beiden Simulationssysteme werden Transformationsregeln im Rahmen der Arbeit betrachtet. Untersuchungen zur Simulation von Geschäftsprozessen wurden auch mit SIMUL8, Plant Simulation und DesmoJ durchgeführt. In der ersten Version wurde darüber hinaus auch Enterprise Dynamics betrachtet (Petsch et al. 2008, S. 217). Vollständige Transformationsregeln in diese Simulationssysteme liegen aber nicht vor. Gründe liegen in Problem die sich bei diesen Simulationsumgebungen ergeben haben.

2.5.1 Arena

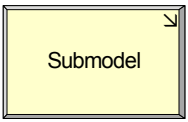
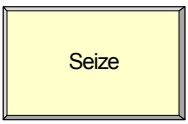
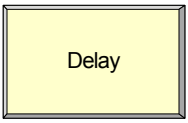
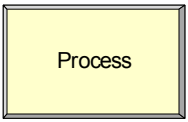
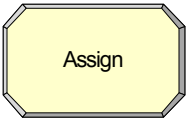
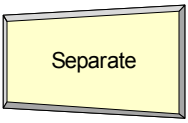
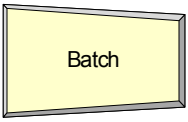
Die Simulationssoftware Arena[10] erlaubt eine grafische Modellierung des Simulationsmodells. Die Komplexität des Simulationsmodells bleibt dem Anwender weitestgehend verborgen. Viele Details des Modells sind aber in den Elementen versteckt und nicht direkt aus dem Modell ersichtlich (Jansen-Vullers und Netjes 2006, S. 88). Ein Element kann daher verschiedene Verhaltensweisen aufweisen, die auf den ersten Blick nicht erkennbar sind. Im Hintergrund wandelt Arena das Modell in SIMAN um. SIMAN verfolgt einen prozessorientierten Ansatz (Kelton et al. 2007, S. 33). Entsprechend ist Arena dem prozessorientierten Simulationsansatz zuzuordnen.

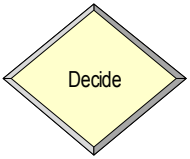
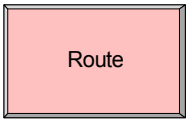
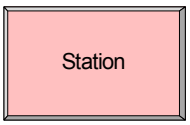
Für die Eignung zur Simulation von Geschäftsprozessen spricht die, von Jansen-Vullers et al. (2006) durchgeführte, Abbildung von Workflow Patterns (van der Aalst et al. 2003). Auf die die Transformationsregeln notwendigen Elemente wird in Tabelle 10 eingegangen.

Tabelle 10: Elemente der Simulationssoftware Arena

Grafische Darstellung	Element	Beschreibung
	Create	Das Create Element erzeugt Marken, die das Simulationsmodell durchlaufen. Es stellt den Anfangspunkt, die Quelle, des Modells dar.
	Dispose	Das Dispose Element entfernt Marken aus dem Simulationsmodell und fungiert als das Ende des betrachteten Modells.

10 Eine Einführung in Arena ist bei Kelton et al. (2007) und Rossetti (2010) zu finden.

Grafische Darstellung	Element	Beschreibung
	Submodel	Das Element Submodel kann als Teilprozess aufgefasst werden. Dieses fasst mehrere Elemente zusammen, die eine logische Einheit bilden können. Jedes Submodel hat genau einen Eingang und einen Ausgang.
	Seize	Das Seize Element reserviert eine Ressource aus dem Ressourcenpool, bindet diese aber nicht fest an die Marke. Steht keine freie Ressource zur Verfügung, dann verharrt die Marke, bis eine Ressource frei wird.
	Release	Das Release Element gibt eine reservierte Ressource frei und stellt diese dem Ressourcenpool zur Verfügung. Die Freigabe kann nur erfolgen, wenn noch mindestens eine Ressource nicht im Ressourcenpool ist.
	Delay	Das Delay Element verzögert eine Marke, bevor diese an das nachfolgende Elemente übergeben wird.
	Process	Das Process Element stellt eine frei definierbare Funktion über die drei Elemente Seize, Delay und Release dar; Seize und Release sind hierbei optional.
	Assign	Das Assign Element erlaubt das Setzen von ein oder mehreren globalen Variablen oder Attributen sowie die Änderung des Typs einer Marke.
	Separate	Das Separate Element kopiert eine Marke. Das Original wird an den ersten ausgehenden Prozesspfad, die Kopien an den zweiten Ausgang geleitet.
	Batch	Ein Batch Element kombiniert eingehende Marken und sendet eine Marke weiter. Durch entsprechende Werte eines Attributs kann ein Original mit dessen erzeugten Kopien verschmolzen werden.

Grafische Darstellung	Element	Beschreibung
	Decide	Das Decide Element stellt einen Entscheidungspunkt im Simulationsmodell dar, mit beliebig vielen ausgehenden Prozesspfaden. Die Entscheidung für einen Prozesspfad ist entweder abhängig von einem Attribut oder basiert auf einer Wahrscheinlichkeit.
	Route	Das Route Element ist einen Sprungpunkt zu einer Station. Es kann, neben der Verknüpfung zwischen den Elementen, mit einer Übertragungszeit hinterlegt werden.
	Station	Das Station Element ist der Zielpunkt eines Route Elements. Mehrere Route Elemente können auf dieses verweisen, womit ab diesem Element der Prozessablauf fortgesetzt wird.

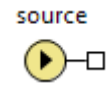
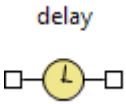
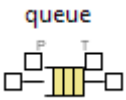
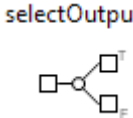
Werden die aufgeführten Elemente grafisch verglichen, so ähneln sich viele von ihnen. Sind die Elemente umbenannt, können diese, visuell, nicht mehr voneinander unterschieden werden. Dieser Umstand muss bei der Validierung des Simulationsmodells beachtet werden.

Eine Besonderheit ist beim Separate Element zu beachten, es gibt nur zwei Ausgänge. Über den ersten Ausgang wird das Original, über den zweiten Ausgang werden die Kopien weitergeleitet. Sollen mehr als zwei Pfade parallel bearbeitet werden, dann sind verschachtelte Separate Elemente notwendig. Ein Batch Element hingegen kann mehrere eingehende Pfade synchronisieren. Ein exklusives Oder gibt es in Arena nur als verzeigendes Element, in Form eines Decide Elements, aber nicht als Zusammenführendes. Zusammengeführt werden die einzelnen Pfade durch eine direkte Verknüpfung mit dem nachfolgenden Element.

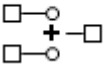
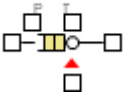
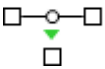
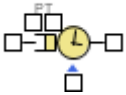
2.5.2 AnyLogic

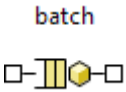
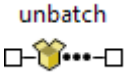
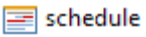
AnyLogic ist eine auf Eclipse[11] aufbauende Simulationsumgebung. Im Hintergrund wird Java verwendet, womit jeder Baustein erweitert werden kann. Simulationsmodelle werden, wie bei Arena, grafisch erstellt. Ebenfalls ist AnyLogic dem prozessorientierten Simulationsansatz zuzuordnen. Die für die Transformationsregeln notwendigen Elemente sind in Tabelle 11 aufgeführt.

Tabelle 11: Elemente der Simulationssoftware AnyLogic

Grafische Darstellung	Element	Beschreibung
	Source	Das Source Element erzeugt als Quelle Marken, die das Simulationsmodell durchlaufen.
	Sink	Das Sink Element entfernt in seiner Funktion als Senke Marken aus dem Simulationsmodell.
	Delay	Ein Delay Element verzögert eine Marke für eine definierte Zeitspanne.
	Queue	Das Queue Element stellt einen Puffer für das nachfolgende Element dar. Erst wenn der Nachfolger frei ist, wird eine Marke an dieses übergeben. Mittels des Queue Elements wird das Verhalten einer Warteschlange abgebildet.
	SelectOutput	Ein SelectOutput stellt einen Entscheidungspunkt im Simulationsmodell dar. Eine Marke wird nur an einen der zwei nachfolgenden Prozesspfade weitergeleitet. Die Entscheidung für den Prozesspfad ist entweder von einem Kriterium oder von einer Wahrscheinlichkeit abhängig.

11 http://www.eclipse.org/

Grafische Darstellung	Element	Beschreibung
selectOutput5 	SelectOutput5	Ein SelectOutput5 stellt einen Entscheidungspunkt mit fünf ausgehenden Prozesspfaden dar.
split 	Split	Ein Split Element erzeugt eine Kopie der Marke. Das Original wird dem oberen Ausgang übergeben, die Kopie dem unteren.
combine 	Combine	Ein Combine Element verbindet zwei eingehende Marken zu einer Marke. Entweder wird eine neue Marke erstellt oder eine eingehende Marke wird weiter verwendet.
resourcePool 	ResourcePool	Ein ResourcePool stellt eine Ressource in ihrer verfügbaren Menge dar.
seize 	Seize	Das Seize Element entnimmt eine Ressource aus dem RecourcePool und weißt diese einer Marke zu. Nur wenn eine Ressource zugewiesen werden kann, wird die Marke weitergeleitet.
release 	Release	Das Release Element gibt eine reservierte Ressource einer Marke frei. Hierbei können nur Ressourcen freigegeben werden, die einer Marke zugewiesen sind.
service 	Service	Das Service Element stellt ein zeitverbrauchendes Element dar. Es kann stelle eine Seize-Delay-Release Kombination dar. Ein Service Element kann eine zusätzlich benötigte Ressource heranziehen und stellt daher eine Aktivität dar.

Grafische Darstellung	Element	Beschreibung
	Batch	Das Batch Element kombiniert mehrere eingehende Marken in einer ausgehenden Marke.
	Unbatch	Das Unbatch Element wandelt eine eingehende Marke in mehrere ausgehende Marken um.
	Schedule	Ein Schedule Element kann als Schichtplan für die Ressourcen sowie zur zeitlichen Steuerung einer Quelle verwendet werden.

Das Split Element kopiert eine Marke. Die Kopie hat keine Verbindung zum Original, ein gegenseitiger Zugriff ist nicht vorgesehen. Werden zwei parallele Pfade mit einem Combine Element zusammengeführt, muss eine Methode definiert werden, um die Attribute der Marken zusammenzuführen. Besonders zu beachten ist dieses Verfahren im Hinblick auf Ressourcen. AnyLogic bindet direkt an eine Marke; Arena entnimmt lediglich aus dem Ressourcenpool. Freigeben kann nur die Marke, die eine Ressource gebunden hat. Im Combine Element muss entweder das Original weiter verwendet werden, oder die Ressource im Rahmen der Methode übergeben werden.

3 Das Transformationsmodell und die Regelbasis

3.1 ProSiT Modell – Process to Simulation Transformation Modell

In der Studie von Hlupic (2000) verwenden 5,5% der akademischen und 11,1% der industriellen Nutzer eine Geschäftsprozesssimulation. Gründe hierfür könnten in den Präferenzen der Benutzer liegen. Akademische Nutzer sehen als wichtigste Punkte bei der Verwendung von Simulation (Hlupic 2000, S. 1679): die Einfachheit der Modellierung sowie die grafische Darstellung und Animation. Für industriellen Nutzer sind die wichtigsten Aspekte (Hlupic 2000, S. 1681): visuelle Darstellung, die Geschwindigkeit sowie die Einfachheit um ein Simulationsmodell zu erstellen.

Diese Aspekte greift der in diesem Kapitel vorgeschlagene Ansatz auf. Die grafische Darstellung wird durch das Transformationsmodell gewährleistet, während die Animation von der Simulationsumgebung unterstützt wird. Die Einfachheit der Modellierung sowie die Geschwindigkeit und Einfachheit zur Erstellung eines Simulationsmodells soll die Regelbasis ermöglichen.

Das Transformationsmodell und die Regelbasis bilden gemeinsam das ProSiT Konzept (Process to Simulation Transformation). Im Kern besteht das Transformationsmodell aus dem ProSiT Ablaufdiagramm. Dieses beinhaltet die Logik des überführten Geschäftsprozesses. Zu diesem werden noch drei Sichten definiert: die Tätigkeitssicht, die Objektsicht und die Ressourcensicht. Die Regelbasis besteht aus drei Teilbereichen. Der erste Teilbereich überführt Geschäftsprozessmodelle in das Transformationsmodell. Die Vorbereitung des daraus erzeugten Modells erfolgt über Normalisierungsregeln. Der dritte Teilbereich überführt das Transformationsmodell in die Simulationsumgebungen.

3.1.1 Das Ablaufdiagramm

Das ProSiT Ablaufdiagramm unterteilt sich in ein konzeptionelles und ein konsistentes Transformationsmodell. Wenn das Ablaufdiagramm durch die Transformationsregeln aus den Geschäftsprozessmodellen erzeugt wurde, handelt es sich um ein konzeptionelles Modell. Durch die Anwendung der Normalisierungsregeln wird ein konsistentes Modell erzeugt. Dieses beinhaltet alle Informationen, die für ein Simulationsmodell notwendig sind. Der Begriff der Normalisierung leitet sich aus der Domäne der Datenmodellierung ab. In dieser wird das Überführen von Datenstrukturen von einer Normalform zur anderen als Normalisierung bezeichnet (Staud 2005, S. 48).

Die Notationselemente des ProSiT Ablaufdiagramms orientieren sich an der BPMN Notation. Auf unterschiede zwischen dem konzeptionellen und konsistenten Modell wird beim jeweiligen Element eingegangen.

Die Aktivität
Das wichtigste Element im ProSiT Ablaufdiagramm ist die Aktivität. Diese leitet sich aus der Definition des Geschäftsprozesses ab. Sie kombiniert Tätigkeit, Objekt und Ressource. Im konzeptionellen Transformationsmodell ist immer die Tätigkeit angegeben und soweit im Geschäftsprozessmodell angegeben, ebenfalls Ressourcen (Abbildung 10).

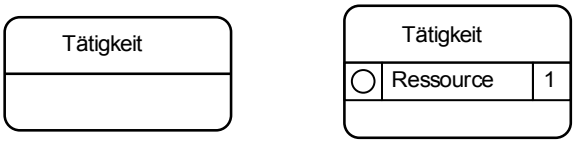


Abbildung 10: ProSiT Ablaufdiagramm – konzeptionelle Aktivität

Die Tätigkeit resultiert aus der Bezeichnung der Aktivität im Geschäftsprozessmodell, beispielsweise „Auftrag anlegen“. Aus der Tätigkeit wird, durch eine Normalisierungsregel, das Objekt der Aktivität ausgelesen.

Für die Aktivität sind drei Marker aus der BPMN Spezifikation definiert[12]: die Schleife, die sequenzielle und parallele Mehrfachverarbeitung (Abbildung 11). Im Zuge der Normalisierung werden die Marker aufgelöst und in andere Notationselemente überführt.

Abbildung 11: ProSiT Ablaufdiagramm – Marker der konzeptionellen Aktivität

Neben den drei Markern gibt es im konzeptionellen Tranformationsmodell auch angeheftete Ereignisse (Abbildung 12). Das linke Ereignis unterbricht die Ausführung der Aktivität und sendet eine Marke an den Nachfolger des Ereignisses. Das gestrichelte Ereignis sendet nur eine Marke, unterbricht die Aktivität aber nicht.

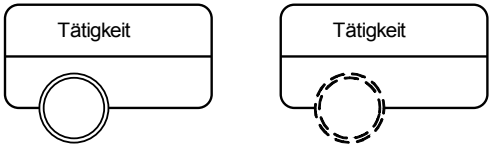


Abbildung 12: ProSiT Ablaufdiagramm – Angeheftete Ereignisse

Die Marker und angehefteten Ereignisse können auch in Kombination auftreten. In diesem Fall betreffen die Ereignisse lediglich die aktuell ausgeführte Aktivität, nicht aber die Mehrfachverarbeitung.

Eine Aktivität wird zur konsistenten Aktivität, wenn diese über alle simulationsrelevanten Informationen verfügt. Hierfür wird die Bearbeitungszeit angeben, weitere Ressourcen hinzugefügt und die Aktivität klassifiziert wird. Eine Aktivität kann hinsichtlich ihrer Bedeutung im Ablaufdiagramm als Hauptaktivität oder als Unter-

12 Vergleiche Abbildung 7 auf Seite 34.

stützungsaktivität und hinsichtlich des Objekts klassifiziert werden. Jeder Prozess weist ein Prozessobjekt auf. Die Tätigkeit der Aktivität bezieht sich entweder auf das Prozessobjekt oder nicht. Beide Eigenschaften können miteinander kombiniert werden (Abbildung 13).

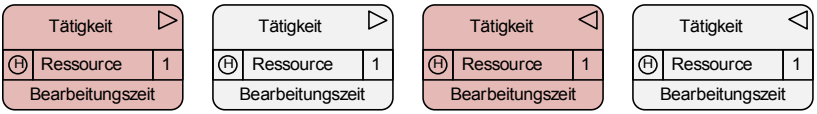


Abbildung 13: ProSiT Ablaufdiagramm – klassifizierte konsistente Aktivität

Die Klassifikation nach Haupt- und Unterstützungsaktivität wird farblich dargestellt, rot eine Haupt-, grau eine Unterstützungsaktivität. Eine am Prozessobjekt ausgeführte Aktivität hat ein nach rechts gerichtetes Dreieck und nach links gerichtet im umgekehrten Fall[13].

Neben den Markern für eine konzeptionelle Aktivität (Abbildung 11) sind zwei Marker für konsistente Aktivität definiert. Verwenden mehrere sequenzielle Aktivitäten die gleichen Ressourcen, können diese in einer kombinierten Aktivität zusammengefasst werden (Abbildung 14 links). Der ursprüngliche Prozessablauf ist dieser dann hinterlegt.

Bei einer parallelen Ausführung in einem Geschäftsprozess kann eine Aktivität vom parallelen Prozesspfad abhängen. Beispielsweise werden die Vitalwerte eines Patienten überwacht und parallel der Patient untersucht. Die Dauer der Überwachung ist abhängig von der Untersuchung. Für die zeitabhängige Aktivität (Abbildung 14 rechts) kann entsprechend keine Bearbeitungszeit hinterlegt werden.

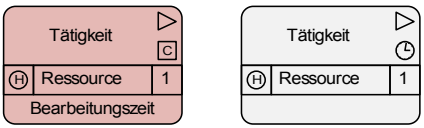


Abbildung 14: ProSiT Ablaufdiagramm – Marker der konsistenten Aktivität

13 Diese Art der Klassifikation richtet sich nach den BPMN Spezifikationen, um bestimmte Sachverhalt hervorzuheben (OMG 2011, S. 41).

Die Bearbeitungszeit der Aktivitäten kann für eine Simulation nicht als konstant angenommen werden. Mögliche Ausprägungen zur Definition von Verteilungen sind in Tabelle 12 aufgeführt. Die Zeiten sind jeweils nach der Struktur [HH:]TT:SS aufgebaut. Für eine Erläuterung der aufgeführten Verteilungen wird auf Kelton et al. (2007, S. 589-602) verwiesen.

Tabelle 12: ProSiT Ablaufdiagramm - Zeitdarstellungen

Verteilung	Muster	Beispiel
Konstant	μ	02:30:00
Normalverteilung	Normal: μ , σ	Normal: 2:00, 0:30
Gleichverteilung	Gleich: a , b	Gleich: 1:00, 4:00
Exponentialverteilung	Expo: α	Expo: 2:00
Gammaverteilung	Gamma: β , α	Gamma: 3:00, 1:00
Dreiecksverteilung	Dreieck: a , m , b	Dreieck: 1:00, 2:00, 4:00
Weibullverteilung	Weibull: α , β	Weibull: 2:00, 1:00

Quelle

Die Quelle baut auf dem Startereignis des BPMN Prozesses auf, das Marken erzeugt, welche das Modell durchlaufen. Hierbei wird eine abweichende grafische Repräsentation (Abbildung 15) verwendet, um relevante Attribute der Quelle darzulegen.

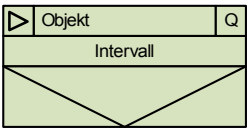


Abbildung 15: ProSiT Ablaufdiagramm – Quelle

Die drei in der Quelle dargestellten Attribute sind das Objekt, die Quantität und das Intervall. Das Objekt entspricht dem Prozessobjekt des Geschäftsprozessmodells und wird automatisch im Rahmen der Normalisierung zugewiesen. Wie viele Marken des Objekts auf einmal erzeugt werden, bestimmt die Quantität. Der zeitliche Abstand zwischen dem Erzeugen von Marken definiert das Intervall, gemäß der Verteilungen in Tabelle 12.

Senke

Die Senke stellt den Endpunkt der Betrachtung oder das Ende eines Geschäftsprozesses dar. Erreicht eine Marke die Senke, wird diese aus dem Modell entfernt. Die grafische Darstellung (Abbildung 16) entspricht der Senke des BPMN Endereignisses (Allweyer 2009, S. 79).

Abbildung 16: ProSiT Ablaufdiagramm – Senke

Alle Marker der Abbildung 7 zum Endereignis in der Senke dargestellt werden. Das Endereignis mit terminierendem Marker stellt aber ein eigenes Element dar.

Terminierende Senke

Die terminierende Senke ist ein Sonderfall der Senke, mit einem anderer semantischen Kontext. Erreicht eine Marke diese Senke, wird die gesamte Instanz beendet. Ihre Darstellung (Abbildung 17) entspricht dem terminierenden Endereignis der BPMN Spezifikation (Allweyer 2009, S. 79). Der semantische Kontext der terminierenden Senke kommt neben der BPMN auch im UML Aktivitätsdiagramm in Form des Aktivitätsendes (OMG 2010, S. 339) vor.

Abbildung 17: ProSiT Ablaufdiagramm – Terminierende Senke

Mehrere zu einer Instant gehörenden Marken, entstehen bei parallelen Bearbeitungen. Diese kommt beispielsweise beim verzweigenden parallelen oder verzweigenden inklusiven Gateway vor. Um die terminierende Senke abzubilden, muss eine Simulationsumgebung alle Marken einer Instanz identifizieren und zeitlichen entfernen können.

Paralleles Gateway

Das parallele Gateway repräsentiert ein logisches Und mit zwei Ausprägungen; als verzweigendes und zusammenführendes Element. Grafische entspricht es (Abbildung 18) dem parallel Gateway der BPMN Spezifikation (OMG 2011, S. 294).

Abbildung 18: ProSiT Ablaufdiagramm – Paralleles Gateway

Das verzweigende parallele Gateway hat einen Vorgänger und mehr als einen Nachfolger. Erreicht eine Marke dieses Element, wird sie vervielfältigt und an alle Nachfolger weitergereicht. Ein zusammenführendes paralleles Gateway verfügt über mehr als einen Vorgänger und einen Nachfolger. Erst wenn von jedem Vorgänger eine Marke der gleichen Instanz das Gateway erreichen, werden diese zusammengefasst und eine Marke an den Nachfolger weitergegeben. Semantisch entspricht ein zusammenführendes paralleles Gateway einer Synchronisation.

Exklusives Gateway

Das exklusive Gateway stellt ein wahrscheinlichkeitsbasiertes exklusives Oder dar und entspricht grafisch (Abbildung 19) dem exklusiven Gateway der BPMN Spezifikation (OMG 2011, S. 291).

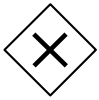


Abbildung 19: ProSiT Ablaufdiagramm – Exklusives Gateway

Das exklusive Gateway ist ebenfalls in ein verzweigendes und ein zusammenführendes Gateway unterteilt. Ein verzweigendes exklusives Gateway benötigt aber ein weiteres Element, den exklusiven Schlüssel (Abbildung 20 links). Es ist ein zusätzliches Element, welches in den Prozessablauf eingebunden wird, aber nicht in den BPMN Spezifikationen vorhanden ist. Der Schlüssel bestimmt die Wahl der auszuführenden Prozesspfade mit einer Angabe der Wahrscheinlichkeit und einer sprachlichen Beschriftung. Diese dient dem Verständnis bei der Validierung des Modells und wird aus dem Quellmodell abgeleitet.

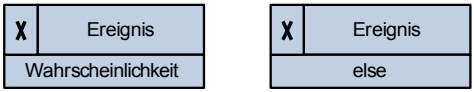


Abbildung 20: ProSiT Ablaufdiagramm – Exklusiver Schlüssel

Jeder Nachfolger eines verzweigenden exklusiven Gateways ist ein exklusiver Schlüssel. Die Summe der Wahrscheinlichkeit alle Schlüssel muss 100% entsprechen oder bei weniger einen exklusiven else-Schlüssel (Abbildung 20 rechts) aufweisen. Der else-Schlüssel leitet sich aus dem BPMN Spezifikationen ab (OMG 2011, S. 291) und wird ausgeführt, wenn kein anderer Pfad aktiviert ist.

Attributsbasiertes Gateway

Das attributsbasierte Gateway entspricht dem exklusiven Gateway, ist aber nicht wahrscheinlichkeitsabhängig (Abbildung 21). Ob ein Prozesspfad auszuführen ist, bestimmt der Wert eines Attributs. Wert und Attribut werden im attributsbasierten Schlüssel (Abbildung 22) definiert.

Abbildung 21: ProSiT Ablaufdiagramm – Attributsbasiertes Gateway

Das Objekt entspricht dem Prozessobjekt und das Attribut ist in der Objektsicht zu definiert. Zwei Bedingungen weist das attributsbasierte Gateway auf: Jeder Schlüssel muss das gleiche Attribut abfragen und die Abfrage muss sich gegenseitig ausschließen. Damit immer ein Prozesspfad ausgeführt wird, ist ein else-Schlüssel für das attributsbasierte Gateway verpflichtend.

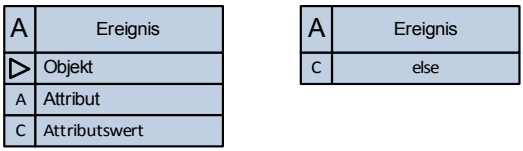


Abbildung 22: ProSiT Ablaufdiagramm – Attributsbasierter Schlüssel

Die Attribute können hinsichtlich verschiedener logischer Operatoren, als Kondition (C) im Schlüssel hinterlegt, abgefragt werden. Eine Übersicht ist in Tabelle 13 aufgeführt.

Tabelle 13: Logische Operatoren im ProSiT Konzept

Kondition	Bedeutung
<	kleiner
<=	kleiner gleich
=	gleich
>=	größer gleich
>	größer
≠	ungleich

Inklusives Gateway

Das inklusive Gateway stammt ebenfalls aus den Spezifikationen der BPMN (OMG 2011, S. 292) und entspricht diesem Notationselement (Abbildung 23). Als inklusives Oder werden bei einem verzweigenden Gateway ein oder mehrere nachfolgende Pfade ausgeführt.

Abbildung 23: ProSiT Ablaufdiagramm – Inklusives Gateway

Das inklusive Gateway ist wahrscheinlichkeitsbasiert, diese wird aber nicht im Schlüssel angegeben. Im Schlüssel wird eine fortlaufende Pfadnummer hinterlegt (Abbildung 24).

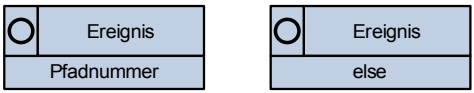


Abbildung 24: ProSiT Ablaufdiagramm – Inklusiver Schlüssel

Die Pfadnummer ist durch zwei Zeichen gekennzeichnet, einem Buchstaben und einer fortlaufenden Nummer, beispielsweise „A1“. Mit dem Buchstaben wird eine Wahrscheinlichkeitstabelle ermittelt. Diese beinhaltet die Wahrscheinlichkeit für jede Pfadkombination, die durch das inklusive Oder ausgeführt werden kann (Tabelle 14).

Tabelle 14: Wahrscheinlichkeitstabelle des inklusiven Gateways

Wahrscheinlichkeitstabelle A	
Pfad	**Wahrscheinlichkeit**
1	10,0%
2	5,0%
3	15,0%
12	25,0%
23	30,0%
123	5,0%
0	10,0%

In der ersten Spalte werden alle Kombinationen der möglichen Prozesspfade aufgeführt. So wird ausschließlich Prozesspfad 1 zu 10% ausgeführt und die Kombination von Prozesspfad 1 und 2 zu 25%. Die Summe aller Wahrscheinlichkeiten muss 100% betragen. Hierdurch wird die inklusive Entscheidung in eine exklusive überführt. Der Pfad 0

ist nur verpflichtend, wenn ein inklusiver else-Schlüssel verwendet wird. In diesem Fall ist der Pfad ebenfalls mit einer Wahrscheinlichkeit anzugeben. Mittels der Wahrscheinlichkeitstabelle können bei der Simulation nur praxisrelevante Pfadkombinationen berücksichtigt werden; im aufgeführten Beispiel entfällt die Kombination 1 und 3.

Nach Ritten (1999, S. 3-4) kann ein inklusives Oder auf drei Arten interpretiert werden. Das ProSiT Konzept unterstützt dieses im konsistenten Transformationsmodell. Ein inklusives Gateway Paar kann hinsichtlich drei Verhaltensweisen klassifiziert werden (Kloos et al. 2009, S. 89): „wait-for-all", „first-come" und „every-time".

Abbildung 25: ProSiT Ablaufdiagramm – Klassifiziertes inklusives Gateway

Im Fall „wait-for-all" (Abbildung 25 links) werden alle ausgelösten Prozesspfade synchronisiert; der Standardfall, wenn das Gateway nicht klassifiziert ist. Beim Fall „first-come" (Abbildung 25 mitte), wird nur die erste angekommende Marke weitergegeben. Spätere Marken werden aus dem Modell entfernt beim Erreichen des zusammenführenden Gateways entfernt. Weitergereicht wird jede Marke im dritten Falle „every-time", wodurch sich die Anzahl der Marken im System erhöht.

Komplexes Gateway

Das komplexe Gateway orientiert sich am attributsbasierten und dem inklusiven Gateway. Darstellung (Abbildung 26) und semantische Logik entsprechen dem komplexen Gateway der BPMN Spezifikation (OMG 2011, S. 295).

Abbildung 26: ProSiT Ablaufdiagramm – Komplexes Gateway

Der komplexe Schlüssel beinhaltet die Logik, nach der ein Prozesspfad eines verzweigenden komplexen Gateways ausgeführt wird. Im Gegensatz zum attributbasierten Schlüssel können beliebige und mehrere Attribute abgefragt werden (Abbildung 27).

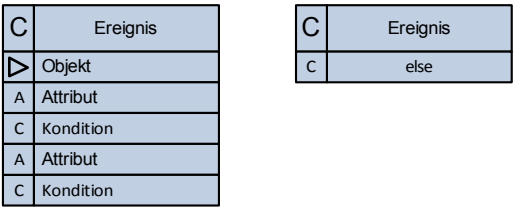


Abbildung 27: ProSiT Ablaufdiagramm – Komplexer Schlüssel

Entsprechend Tabelle 14 kann für jedes Attribut eine andere Konditionsart verwendet werden. Alle Konditionen müssen erfüllt sein, damit ein Prozesspfad ausgeführt wird. Damit immer ein Prozesspfad ausgeführt wird, ist ein else-Schlüssel verpflichtend.

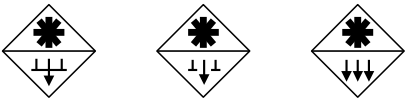


Abbildung 28: ProSiT Ablaufdiagramm – Klassifiziertes komplexes Gateway

Das zusammenführende Gateway bietet die gleichen Ausprägungen wie das inklusive Gateway. Die drei Fälle sind in Abbildung 28 enthalten.

Verzögerung

Eine Verzögerung stellt eine Liege- oder Übermittlungszeit im Prozessmodell dar. Das Element (Abbildung 29) orientiert sich am Timer Event der BPMN Spezifikationen (OMG 2011, S. 251).

Abbildung 29: ProSiT Ablaufdiagramm – Verzögerung

Das Attribut Zeit kann mit den Zeitmustern aus Tabelle 13 versehen werden.

Ressourcenbindung

Die Ressourcenbindung bindet eine Ressource an eine Marke. Die Ressource bleibt nach Abschluss einer Aktivität an der Marke gebunden und wird nicht freigegeben. Die grafische Darstellung ist in Abbildung 30 aufgeführt.

Abbildung 30: ProSiT Ablaufdiagramm – Ressourcenbindung

Das Attribut bestimmt die Ressource und das „Q" die Anzahl. Eine Ressource, die für einen Prozesspfad reserviert ist, steht allen Aktivitäten in diesem zur Verfügung. Die Ressource muss nicht erneut reserviert werden. Ist keine Ressource frei, wartet die Marke in einer unbegrenzten „First in First out" Warteschlange.

Ressourcenfreigabe

Das Gegenstück zur Ressourcenbindung ist die Ressourcenfreigabe. Diese gibt eine reservierte Ressource frei in der angegebenen Menge frei (Abbildung 31).

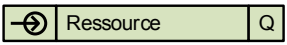


Abbildung 31: ProSiT Ablaufdiagramm – Ressourcenbindung

Instanziierung

Die Instanziierung erzeuge mehrere Marken innerhalb eines Prozesspfades. Semantisch entspricht diese einem logischen Und, aber mit nur einem eingehenden und einem ausgehenden Prozesspfad. Mit diesem Element kann eine Aktivität mehrfach in einer Instanz ausgeführt werden. Die verzweigende Instanziierung (Abbildung 32 links) erzeugt die Kopien der Marke und die zusammenführende Instanziierung (Abbildung 32 rechts) führt diese zusammen.

Abbildung 32: ProSiT Ablaufdiagramm – Instanziierung

Das Objekt für die Instanziierung wird aus dem Prozessobjekt abgeleitet. Synchronisiert werden beim Zusammenführen nur die Marken, die zur gleichen Instanz gehören.

Attributsfestlegung

Das Element Attributsfestlegung ändert Attribute einer Marke (Abbildung 33). Das Attribut einer Marke kann für ein attributsbasiertes oder komplexes verzweigendes Gateway verwendet werden sowie für die Auswertung einer Simulationsstudie.

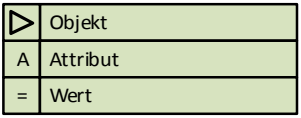


Abbildung 33: ProSiT Ablaufdiagramm – Attributsfestlegung

Damit ein Attribut verwendet werden kann, muss dieses in der Objektsicht definiert werden.

Teilprozess

Der Teilprozess ermöglicht es, ein detaillierteres Modell aufzurufen. Das Element orientiert sich am Teilprozess in der BPMN Spezifikation (OMG 2011, S. 174), entspricht semantisch jedoch einem hinterlegten Prozess bei einer Funktion der eEPK. Der Teilprozess ist in Abbildung 34 dargestellt.

Abbildung 34: ProSiT Ablaufdiagramm – Teilprozess

Ein Teilprozess kann im konzeptionellen Transformationsmodell mit Markern charakterisiert werden. Im Zuge der Normalisierung wird die semantische Bedeutung in andere Elemente überführt. Abbildung 35 führt alle Marker auf, welche vom Teilprozess unterstützt werden.

Abbildung 35: ProSiT Ablaufdiagramm – Marker beim konzeptionellen Teilprozess

Mit Ausnahme des vierten Markers sind die ersten drei auch für die konzeptionelle Aktivität definiert. Beim ersten Marker, der Schleife, ist die Anzahl der Ausführungen nicht bekannt. Bei der sequenziellen und parallelen Mehrfachverarbeitung ist beim Start der Aktivität die Anzahl der Ausführungen bereits festgelegt. Der vierte Marker steht für eine Ad-Hoc Bearbeitung. Im Ad-Hoc Teilprozess ist nur Aktivitäten erlaubt (OMG 2011, S. 182), die in keiner Beziehung zueinander stehen.

Ein Teilprozess hat einen Vorgänger und einen Nachfolger. Entsprechend benötigt der Teilprozess jeweils ein Element, um die enthaltene Abfolge mit dem Vorgänger und Nachfolger des Teilprozesselements zu verknüpft. In Analogie zur BPMN Spezifikation wird das Start- und Endereignis mit einem Plus-Marker versehen (Abbildung 36).

Abbildung 36: ProSiT Ablaufdiagramm – Verbindungselemente

Wie bei der Aktivität können Ereignisse an den Teilprozess angeheftet werden (Abbildung 37). Erlaubt ist jedoch nur das unterbrechende Ereignis, um die Transaktion aus den BPMN Spezifikationen (OMG 2011, S. 179) abzubilden. Ein nicht unterbrechendes angeheftetes Ereignis ist in Anlehnung an BPMN nicht spezifiziert. Das Ereignis selbst kann mit den Marken aus Abbildung 7 versehen werden. Ein Teilprozess muss dann aber ein entsprechend markiertes Endereignis aufweisen, um die Verknüpfung herstellen zu können.

Abbildung 37: ProSiT Ablaufdiagramm – Marker beim konzeptionellen Teilprozess

Sprungpunkt

Der Sprungpunkt ist ein weiteres Element, zwei Modelle miteinander zu verknüpfen, die sich auf gleicher Hierarchiestufe befinden. Neben dem Sprungpunkt (Abbildung 38 links) gibt es den Zielpunkt (Abbildung 38 rechts).

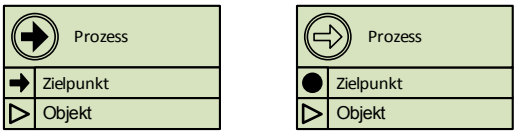


Abbildung 38: ProSiT Ablaufdiagramm – Sprung- und Zielpunkt

Semantisch orientiert sich der Sprungpunkt an der Prozessschnittstelle der eEPK. Mit den Attributen Prozess und Zielpunkt wird im Sprungpunkt der Zielpunkt verlinkt. Ein Zielpunkt kann daher von mehreren Sprungpunkten angesteuert werden. Das Objekt in beiden Elementen entspricht dem Prozessobjekt des jeweiligen Prozesses. Unterscheidet sich das Objekt zwischen Sprung- und Zielpunkt, dann liegt ein Objektwechsel vor; gleich benannte Attribute werden aber automatisch übernommen. Im weitesten Sinne stellt ein Zielpunkt eine Art Quelle im ProSiT Ablaufdiagramm dar.

3.1.2 Die Tätigkeitssicht

Die Tätigkeitssicht fokussiert die Tätigkeit in den Aktivitäten des ProSiT Ablaufdiagramms. Kern der Tätigkeitssicht ist ein domänenabhängiges Repository (Kloos et al. 2010; Kloos et al. 2011).

Das Repository der Tätigkeitssicht betrachtet ausschließlich Verben und wird für die Klassifikation und die Festlegung der Zeiterfassungsmethode für Aktivitäten verwendet. Das Repository ist domänenabhängig konzipiert, da Verben in verschiedenen Domänen unterschiedliche Bedeutungen haben. Das Verb „drucken“ im Gesundheitswesen deutet beispielsweise auf eine unterstützende Aktivität hin, während es in einer Druckerei den Hauptaktivitäten zuzuordnen wäre.

Die zentrale Tabelle (15) des Repositorys enthält die Verben. Zusätzliche Informationen werden über weitere Tabellen hinzugefügt. Ein Verb kann, muss allerdings nicht, in mehr als einer Domäne definiert werden.

Tabelle 15: Tätigkeitssicht – Verbtabelle

Domäne	Verb
D1	V1
D1	V2
D1	V3
D2	V2
D2	V4

Aus dem Prozesstyp leitet sich Tabelle 16 ab. Diese beinhaltet die Klassifikation nach Kern-, Unterstützungs- und Führungsprozess.

Tabelle 16: Tätigkeitssicht – Prozessarttabelle

Prozessart	Bezeichnung
KP	Kernprozess
UP	Unterstützungsprozess
FP	Führungsprozess

Eine zweite Kontrolltabelle (17) ist die Aktivitätsart definiert. Sie beinhaltet die Klassifikationen Haupt- und Unterstützungsaktivität.

Tabelle 17: Tätigkeitssicht – Aktivitätsklassifikation

Aktivitätsart	Bezeichnung
HA	Hauptaktivität
UA	Unterstützungsaktivität

Teile der Normalisierung bauen auf diesen beiden Klassifikationen auf. Alle Klassifikationen werden in der Klassifikationstabelle (18) zusammengefasst. Die Aktivitätsart sowie die Ausführung am Prozessobjekt stellen nur einen Vorschlag dar, da die verwendeten Verben vom Modellierer oder den Modellierungsrichtlinien abhängen (Kloos et al. 2010, S. 96).

Tabelle 18: Tätigkeitssicht – Klassifikationstabelle

Domäne	Verb	Prozessart	Aktivitätsart	Prozessobjekt
D1	V1	KP	HA	ja
D1	V1	UP	HA	ja
D1	V2	KP	HA	nein
D1	V2	UP	UA	ja
D1	V2	FP	UA	ja
D2	V2	KP	UA	ja
D2	V2	UP	UA	nein

Wie aus der beispielhaften Darstellung in Tabelle 18 hervorgeht, sind die Aktivitätsart und die Anwendung am Prozessobjekt voneinander unabhängig. Da Verben sprachliche Elemente sind, können für diese auch Synonyme verwendet werden (Friedrich 2009). Hierfür ist eine Synonymtabelle (19) definiert, um den Pflegeaufwand der Tätigkeitssicht zu reduzieren. Um domänenabgängige Synonyme berücksichtigen zu können, ist die Synonymtabelle domänenabgängig definiert.

Tabelle 19: Tätigkeitssicht – Synonymtabelle

Domäne	Synonym	Verb
D1	S1	V1
D1	S2	V1
D1	S3	V2
D1	S4	V3
D2	S4	V2
D2	S5	V3

Neben der Klassifikation der Aktivitäten empfiehlt das Repository eine Erhebungsmethode für die Bearbeitungszeit von Aktivität. Die Empfehlung richtet sich nach dem Verb, der Prozessart und der Aktivitätsart (Kloos et al. 2011, S. 29). Eine schematische Darstellung bietet Tabelle 20.

Tabelle 20: Empfehlungsmatrix für die Erfassungsmethode

Verb	Kernprozess		Unterstützungsprozess		Führungsprozess	
	HA	UA	HA	UA	HA	UA
V1	messen	befragen	messen	befragen	befragen	befragen
V2	messen	befragen	messen	befragen	befragen	befragen
V3	befragen	schätzen	befragen	schätzen	befragen	befragen
V4	schätzen	schätzen	schätzen	schätzen	schätzen	schätzen

In Anlehnung an Kloos et al. (2011, S. 29)

Das ProSiT Konzept sieht vier Methoden für die Erfassung der Bearbeitungszeit einer Aktivität vor (Kloos et al. 2011, S. 98): System, messen, befragen und schätzen. Bei der ersten Methode werden bereits vorhandene Zeiten aus Informationssystem (E_{sys}) verwendet. Liegen keine erfassten Zeiten vor, können diese mit mehren Einzelmessungen erfasst werden (E_{mes}) und eine Verteilung der Zeiten abgeleitet werden. Bei der dritten Methode wird eine ausführende Person nach der Bearbeitungszeit befragt (E_{bef}) und bei der vierten Methode erfolgt eine Schätzung durch den Modellierer (E_{sch}). Werden verschiedene Personen befragt oder mehrere Schätzungen abgegeben, kann aus diesen eine einfache Verteilung abgeleitet werden.

Unterschiedliche Methoden sind definiert, da nicht für jede Aktivität ein größerer Aufwand betrieben werden muss, um die Bearbeitungszeit zu ermitteln. Beispielsweise könnten „triviale" Tätigkeiten, wie das Drucken von Untersuchungsergebnissen (Kloos et al. 2011, S. 98) geschätzt werden, während die eigentliche Untersuchung gemessen wird.

Bei den vier Methoden wird von einer absteigenden Sicherheit der erfassten Zeiten ausgegangen. So sind gemessene Zeiten realitätsgetreuer, als eine geschätzte Zeit des Modellierers.

$$E_{sys} > E_{mes} > E_{bef} > E_{sch} \quad (1)$$

Gemäß Formel (1) wären erfasste Zeiten aus einem Informationssystem dem Messen zu bevorzugen, sowie das Messen dem Befragen.

Hierbei gilt die Annahme, dass die Zeiten aus dem Informationssystem korrekt erfasst wurden. Da ein Informationssystem nicht vorausgesetzt werden kann, gibt es im ProSiT Konzept nur drei empfohlene Erfassungsmethoden (Tabelle 21).

Tabelle 21: Tätigkeitssicht – Erfassungsmethodentabelle

Erfassungsmethode	Bezeichnung
MES	messen
BEF	befragen
SCH	schätzen

In Tabelle 22 sind die Verben V1 und V2 aus Tabelle 20 übernommen, um die Darstellung zu visualisieren. Daher wird in der Spalte nicht der Schlüssel der Erfassungsmethode aufgeführt, sondern die Bezeichnung, damit ein direkter Vergleich mit Tabelle 20 erfolgen kann.

Tabelle 22: Tätigkeitssicht – Erfassungsmethodenvorschlagstabelle

Domäne	Verb	Prozessart	Aktivitätsart	Methode
D1	V1	KP	HA	messen
D1	V1	KP	UA	befragen
D1	V1	UP	HA	messen
D1	V1	UP	UA	befragen
D1	V1	FP	HA	befragen
D1	V1	FP	UA	befragen
D1	V2	KP	HA	messen
D1	V2	KP	UA	befragen
D1	V2	UP	HA	messen
D1	V2	UP	UA	befragen
D1	V2	FP	HA	befragen
D1	V2	FP	UA	befragen

Mit Tabelle 22 ist das konzipierte Repository dargelegt, welches durch die Tätigkeitssicht definiert wird. Die Klassifikation der Aktivität sowie die Zuweisung der Erfassungsmethode für die Bearbeitungszeit erfolgt im Rahmen der Normalisierung.

3.1.3 Die Objektsicht

Die Objektsicht fokussiert auf Objekte, die im Ablaufdiagramm verwendet werden. Bei den vier Elementen Quelle, attributsbasiertes Gateway, komplexes Gateway und Attributsfestlegung wird es durch das Prozessobjekt bestimmt. Bei der Aktivität hingegen gibt es eine Wahl bezüglich der Ausführung am Prozessobjekt. Im Rahmen der Objektsicht wird dieser Aspekt aber nicht unterschieden.

In der Objektsicht müssen nur die Attribute von Objekten gepflegt werden, die im Rahmen der Prozessmodelle verwendet werden. Zusätzliche Attribute können gepflegt werden. Im ProSit Ablaufdiagramm ist es aber nur vorgesehen, über die Attributsfestlegung Attribute zu ändern.

Zur Definition von Objekten wird ein vereinfachtes UML Klassendiagramm verwendet, in dem nur Attribute definiert werden. Methoden, Assoziationen oder Vererbung (Kecher 2009, S. 31) werden nicht berücksichtigt. Ein Beispiel für ein Klassendiagramm in der Objektsicht ist in Abbildung 39 dargestellt.

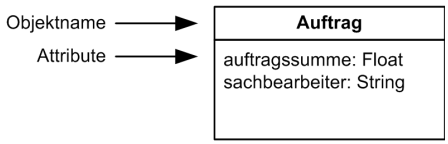


Abbildung 39: Objektsicht – Klassendiagramm

Das dargestellte Klassendiagramm wird in einer dreispaltigen Tabelle 23 verwaltet, bestehend aus den Spalten: Objektname, Attributsname und Attributstyp.

Tabelle 23: Verwaltung der Objekt in der Objektsicht

Objektname	Attributsname	Attributstyp
Auftrag	auftragssumme	Float
Auftrag	sachbearbeiter	String
Patient	krankheit	String
Patient	brillenträger	Boolean

Die definierten Objekte werden über Transformationsregeln in die Simulationsumgebungen überführt. Transformationsregeln werden aber nur für Simulationsumgebungen definiert, die eine explizierte Definition der Attribute fordern. So müssen bei AnyLogik Attribute in Klassen definiert werden, damit Marken auf diese zugreifen können. Bei Arena können Attribute hingegen frei belegt und geprüft werden, ohne diese zuvor zu definieren.

Die Pflege der Objekte kann auf zwei Varianten erfolgen. Wenn die Objektsicht gepflegt ist, kann das Prozessobjekt für die Ablaufdiagramme aus einer Liste ausgewählt werden. Sind jedoch bereits die Prozessobjekte bei den Ablaufdiagrammen hinterlegt, können diese extrahiert werden und als Vorlage für die Objektsicht verwendet werden. Ebenfalls können Attribute extrahiert werden, die in der Attributsfestlegung oder Gateways verwendet werden. In diesem Falle müssen anschließend nur die Attributstypen festgelegt werden, um die Pflege der Objektsicht abzuschließen.

3.1.4 Die Ressourcensicht

Die Ressourcensicht definiert die im ProSiT Ablaufdiagramm verwendeten Ressourcen. Bevor der Aufbau der Ressourcensicht erläutert wird, erfolgt eine kurze Betrachtung von Ressourcen im Kontext der Simulation von Geschäftsprozessen. So untersuchen van der Aalst et al. (2010) Einschränkungen von bestehenden Ansätzen, welche die Simulation als Werkzeug für die Analyse von Geschäftsprozessen verwenden. Die von den Autoren untersuchten Simulationsansätze zielen auf die Erstellung des Simulationsmodells ab. Ein ermitteltes Problem ist die „Reißbrettmodellierung" des Simulationsmodells, anstatt die Verwendung von bestehenden Informationsquellen. Unter der Annahme Geschäftsprozesse werden nach den Vorgaben ausgeführt, wird dieses Problem durch die Transformation von Geschäftsprozessmodellen umgangen. Kern der Untersuchung ist jedoch die Verfügbarkeit der Ressourcen. Vier Ressourcenprobleme wurden herausgearbeitet (van der Aalst et al. 2010, S. 319-320):

- Menschen sind in mehreren Prozessen involviert
- Menschen arbeiten nicht mit einer konstanten Geschwindigkeit
- Menschen tendieren zur Teilzeitarbeit und führen die Arbeit in Bündeln aus
- Prozesse können sich abhängig vom Kontext ändern

Bei den von van der Aalst et al. (2010) untersuchten Simulationsansätzen hat jede Aktivität zugewiesene Ressourcen und jede Ressource steht in einer festen Anzahl zur Verfügung. Darüber hinaus wird eine Vollzeitverfügbarkeit dieser Ressourcen unterstellt. In der Realität kann eine Ressource in 10 verschiedenen Prozessen involviert sein und spendet 20% der verfügbaren Zeit in den untersuchten Prozess. Eine

Vollzeitverfügbarkeit kann aufgrund dieses Sachverhalts nicht als realitätsnah bezeichnet werden.

Diese aufgeführten Probleme werden im ProSiT Konzept adressiert. Ist ein Mensch in mehreren Prozessen involviert, müssen diese Prozesse für die Simulation herangezogen werden. Wird nur ein Prozess berücksichtigt, muss die Verfügbarkeit der Ressourcen mittels eines Schichtplans oder eines zusätzlichen Prozesses berücksichtigt werden. Ein Beispiel hierfür zeigt Allweyer (2005, S. 247); der zusätzliche Prozess besteht aus einer Quelle, einer Aktivität und einer Senke. Die nicht konstante Geschwindigkeit der Bearbeitungszeiten kann über eine Verteilung realisiert werden. Prozesse, die sich über die Zeit ändern, können nur so lange simuliert werden, solange diese gültig sind. Diese aufgeführten Probleme zu vermeiden, obliegt der Verantwortung des Modellierers.

Das Erledigen vorliegender Arbeit als Bündel wird vom ProSiT Konzept nicht adressiert. Eine mögliche Realisierung wäre, eine Warteschlange, die für einen Zeitraum alle ankommenden Marken sammelt und diese auf einmal an die nachfolgende Aktivität weiterleitet. Dieses semantische Konstrukt muss jedoch von einer Simulationsumgebung unterstützt werden. Eine andere Möglichkeit wäre die Verwendung des Instanziierungselements des ProSiT Ablaufdiagramms. Mehrere Marken werden zunächst in einer Marke zusammengefasst und anschließend durch ein darauf folgendes Instanziierungselement wieder in mehrere Marken zerteilt. Die darauf folgende Aktivität benötigt die höchste Priorität, damit alle Marken auf einmal abgearbeitet werden. Durch diesen Mechanismus werden aber getrennte Instanzen aufgelöst, sodass es zu Problemen bei der Synchronisation eines zusammenführenden parallelen Gateways kommen kann. Dieser von van der Aalst et al. (2010, S. 320) aufgeführte Punkt weißt weiteren Forschungsbedarf auf.

Zur Klassifikation von Ressourcen

Wie aus van der Aalst et al. (2010) hervorgeht, sind Menschen wichtige Ressourcen, die für die Simulation von Geschäftsprozessen notwendig sind. Wird eine eEPK betrachtet, wird dies bestätigt; Personen werden durch Stellen oder Rollen repräsentiert. Neben Menschen können noch andere Ressourcen notwendig sein. Zur Klassifikation der möglichen Ressourcen wird ein aus vier Elementen bestehender morphologischer Kasten (Tabelle 24) verwendet.

Tabelle 24: Klassifikation der Ressourcen

Eigenschaft	**Ausprägungen**		
Beweglichkeit	Mobil	Stationär	Immateriell
Beschaffenheit	Menschlich	Nicht-Menschlich	
Bestand	Konsumiert	Konsistent	
Bearbeitung	Ausführend	Unterstützend	

Die Eigenschaft der Beweglichkeit hat drei Ausprägungen. Die ersten beiden – mobil und stationär – beziehen sich auf die physische Welt. Eine Ressource kann entweder den physischen Aufenthaltsort ändern (Mensch) oder verharrt am gleichen Ort (Raum). Ein Sonderfall bei der Beweglichkeit ist die immaterielle Ausprägung. Immateriellen Ressourcen liegt die Annahme zugrunde, sie sind an verschiedenen Orten zeitlich verfügbar. Aufgrund dieser Annahme können diese nicht als simulationsrelevante Ressourcen aufgefasst werden. Da kein Engpass entstehen kann, werden immaterielle Ressourcen nicht betrachtet. Heß und Meis (2011, S. 101) schließen diese Ressourcen ebenfalls in ihrem konzipierten Pflegedienstleistungsmodell aus; wobei die Autoren nur „Humanressourcen“ und „physische Ressourcen“ unterscheiden.

Dieser Aspekt wird durch die zweite Eigenschaft, die Beschaffenheit, adressiert. Sie unterscheidet zwischen menschlichen und nicht-menschlichen Ressourcen. Menschen stehen in der Regel nur für eine begrenzte Zeitspanne zur Verfügung, die mittels Schichtplan

beschrieben werden kann. Bei nicht-menschlichen Ressourcen ist dies nicht zwingend der Fall. Ein Raum könnte aber auch mit einem Schichtplan versehen werden, wenn dieser außerhalb des betrachteten Simulationskontexts verwendet wird.

Eine weitere Eigenschaft von Ressourcen ist ihre Beständigkeit. Wird eine Ressource für eine Aktivität herangezogen, kann diese entweder konsumiert werden oder bleibt bestehen. Durch den Konsum steht die Ressource für andere Aktivitäten nicht mehr zur Verfügung, beispielsweise bei Rohstoffen im Rahmen einer Produktion. Bei Geschäftsprozessen ist diese Ausprägung nicht von Bedeutung. So werden in Geschäftsprozessmodellen nur konsistente Ressourcen verwendet.

In welcher Weise eine Aktivität von einer Ressource ausgeführt wird, beschreibt die vierte Eigenschaft, die Bearbeitung. Entweder führt eine Ressource die Aktivität aus oder sie unterstützt die Ausführung. Aus sprachlicher Sicht wird eine ausführende Ressource mit einem Nominativ beschrieben, unterstützende mit Dativ sowie Genitiv. Auswirkungen auf eine Simulationsstudie hat diese Unterscheidung nicht; das ProSiT Konzept berücksichtigt diesen Aspekt daher nicht. Diese Unterscheidung stellt jedoch einen Forschungspunkt dar, bei dem die Auswirkungen auf Normalisierungsregeln untersucht werden können, insbesondere in Richtung der Erhebung relevanter Bearbeitungszeiten.

Aus dem morphologischen Kasten leiten sich vier Arten von Ressourcen ab, die im ProSiT Konzept verwendet werden. Aus der Beschaffenheit kommen die Menschen. Diese sind zwangsläufig mobil und konsistent. Die restlichen Drei ergeben sich aus der Eigenschaft Beweglichkeit. Zum einen gibt es stationäre Ressourcen, beispielsweise Räume zum anderen bewegliche Ressourcen wie ein Bett in einem Krankenhaus. Diese beiden Ressourcen sind jeweils konsistent. Als dritte Ressourcenart können immaterielle Ressourcen aus der Eigenschaft Beweglichkeit abgeleitet werden. Immaterielle Ressourcen

sind an sich immer konsistent, wobei auch konsumierende immaterielle Ressourcen in Hinblick auf die Verfügbarkeit bestimmter Dienste denkbar wären. Immaterielle Ressourcen werden aber in der gegenwärtigen Konzeption nicht im ProSiT Konzept berücksichtigt. Diese werden auch als offener Forschungspunkt für das ProSiT Konzept behandelt.

Gestaltung der Ressourcensicht

Das Modell der Ressourcensicht orientiert am UML Klassendiagramm. Als zentrale Klasse wird die Ressource verwendet. Von dieser leiten sich der Mensch, die bewegliche und die stationäre Ressource ab. Einer Ressource wird im Rahmen der Normalisierung einer dieser drei Typen zugeordnet. Die Grundstruktur (Abbildung 40) definiert nur zwei Attribute – die verfügbare Menge und ein Schichtplan – um die Kompatibilität zu den einzelnen Simulationsumgebungen sicherzustellen.

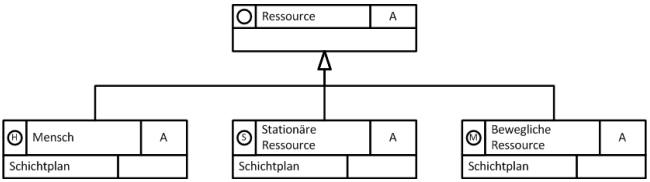


Abbildung 40: Grundstruktur der Ressourcensicht

Alle vier dargestellten Ressourcen sind als abstrakt (A) gekennzeichnet. Anstelle dieses Markers wird in den konkreten Ressourcen die verfügbare Menge angegeben. Wird ein Schichtplan verwendet, so ist die Angabe die Standardmenge, die durch den Schichtplan überschrieben wird. Aus Sicht der Vererbung könnte das Attribut Schichtplan jedoch kritisiert werden. Dieses wird bei allen erbenden Klassen verwendet und sollte bei der Mutterklasse verwendet werden. Dies erfolgt aufgrund des Gedankens der Weiterentwicklung nicht. Werden immaterielle Ressourcen berücksichtigt, wie bei Kloos et al. (2009, S. 90) aufgeführt, so verfügen diese über keinen Schichtplan.

Weitere mögliche Attribute könnten Kosten oder die Geschwindigkeit sein. Kosten können fix und variable sein, sind jedoch in Arena direkt umsetzbar. Die Geschwindigkeit, die für bewegliche Ressourcen berücksichtigt werden könnte, wird in Arena nicht unterstützt und in AnyLogic nur, wenn eine Netzwerkmodellierung verwendet wird. Das ProSiT Konzept arbeitet jedoch nicht mit den Netzwerkmodellierungsansatz, weshalb dieses Attribut ebenfalls nicht direkt umgesetzt werden kann.

Das ProSiT Ablaufdiagramm bietet bezüglich der Geschwindigkeit und stationären Ressourcen eine Weiterentwicklungsmöglichkeit. Bei einer Aktivität ist nur eine stationäre Ressource vorgesehen. Zwei stationäre Ressourcen in einer Aktivität müssen sich am gleichen Ort befinden. Durch den Wechsel des physischen Ortes muss die Marke aber auch bewegliche Ressourcen eine Wegstrecke überbrücken. Wird dieser Aspekt berücksichtigt, kann ein realitätsgetreueres Simulationsmodell erstellt werden.

Ressourcengruppen werden ebenfalls nicht von der Ressourcensicht des ProSiT Konzepts als Ganzes unterstützt. Diese fassen einzelne Ressourcen zusammen, die unterschiedlich verwendet werden. Ein Beispiel hierfür sind Räume. Alle Räume können für eine Voruntersuchung verwendet werden, aber nur zwei könnten für weiterführende Untersuchungen verfügbar sein. Um dieses Verhalten zu ermöglichen, müssen spezielle Transformationsregeln konzipiert werden. Ebenfalls müssen Ressourcengruppen von Simulationsumgebungen unterstützt werden, wie dies mit Arena der Fall ist.

Als offener Punkt in der Ressourcensicht ist noch der Schichtplan zu definieren. Dieser wird in Anlehnung an die betrachteten Simulationsumgebungen als sich wiederholende Zeiträume aufgefasst, bei denen eine unterschiedliche Menge an Ressourcen zur Verfügung steht. Ein Beispiel hierfür ist exemplarisch in Tabelle 24 aufgeführt.

Tabelle 25: Beispiel eines Schichtplans der Ressourcensicht

Zeitraum	Menge
00:00:00:00 – 01:00:00:00	5
01:00:00:00 – 02:00:00:00	6
02:00:00:00 – 03:00:00:00	4
03:00:00:00 – 04:00:00:00	5
04:00:00:00 – 05:00:00:00	5
05:00:00:00 – 06:00:00:00	0
06:00:00:00 – 07:00:00:00	0

Ein Schichtplan wird für unterschiedliche Zeiträume definiert. Hierbei wird das Muster aus Tabelle 12 für die Definition einer Bearbeitungszeit eingehalten. Wenn die angegebenen Zeiträume durchlaufen sind, wird der Schichtplan wieder von vorne begonnen. Bei einer softwaretechnischen Umsetzung kann anstelle der Tabelle auch ein , wie bei Arena, verwendet werden.

3.2 Das ProSiT Konzept und dessen Regelbasis

Das ProSiT Konzept bildet sich auf dem ProSiT Modell und der Regelbasis. Ein Quellmodell wird in das ProSiT Transformationsmodell überführt. Aus diesem Schritt resultiert konzeptionelle Transformationsmodell, dass sich begrifflich aus Pages Vorgehensmodell ableitet (Abbildung 42). Um das Transformationsmodell in eine Simulationsumgebung zu überführen, muss dieses mit simulationsrelevanten Daten angereichert werden sowie hinsichtlich des Detaillierungsgrads angepasst werden. Für diese Schritte wird im ProSiT Konzept der Begriff der Normalisierung verwendet. Dieser leitet sich aus der Domäne der Datenmodellierung ab, in der Datenstrukturen von einer Normalform zu einer anderen mittels Normalisierung überführt wird (Staud 2005, S. 48). Aus einem normalisierten konzeptionellen Transformationsmodell entsteht das konsistente Transformationsmodell. Dieses beinhaltet alle notwendigen

Daten und Strukturen, um simuliert werden zu können. Das konsistente Transformationsmodell wird mittels Transformationsregeln in eine passende Simulationsumgebung überführt.

Im ersten Teilbereich, der Transformation von Quellmodellen in das ProSiT Transformationsmodell sind drei verschiedene Regelarten vorzufinden: Syntax-, Vorbereitungs- und Transformationsregeln. Syntaxregeln prüfen, ob ein Modell syntaktisch richtig modelliert ist. Liefern die Syntaxregeln ein positives Ergebnis, können die Vorbereitungsregeln angewendet werden. Diese entfernen nicht benötigte Elemente oder vereinfachen Strukturen. Das vorbereitete Quellmodell wird anschließend mittels der Transformationsregeln in das Transformationsmodell überführt. Das resultierende Modell wird als konzeptionell bezeichnet, da nur die Informationen vorhanden sind, die aus den Quellmodellen überführt wurden.

Syntaxregeln legen lediglich die Annahmen fest, die Bedingungen die ein Modell erfüllen muss, damit eine Transformation erfolgen kann. Die eigentliche Transformation erfolgt über die Vorbereitungs- und Transformationsregeln. Dieses zweistufige Transformationskonzept orientiert sich an Hoyer et al. (2007, S. 190), wie dies erstmals in Kloos (2010, S. 92) beziehungsweise Kloos (2011, S. 25) aufgeführt. Hoyer et al. (2007) beschreiben in ihrer Arbeit die Überführung einer privaten eEPK zu einem öffentlichen BPMN Prozess. Hierbei werden im ersten Schritt Abstraktionsregeln (Im ProSiT Konzept die Vorbereitungsregeln) auf der eEPK angewendet, um Informationen aus dem Modell zu entfernen, die nicht für Geschäftspartner sichtbar sein sollen. Anschließen werden auf diesem abstrahierten Modell Mapping Regeln (Im ProSiT Konzept die Transformationsregeln) angewendet, welche die Transformation in einen öffentlichen BPMN Prozess umfassen.

Als eine Herausforderung führen Hoyer et al. (2007, S. 193) Eingriffe des Benutzers auf, um Ereignisse der EPK in den BPMN Prozess zu überführen. Eine automatische Transformation sei nicht möglich. Bei der EPK gibt es ein Notationselement für Ereignisse, abhängig vom

Einsatzort entweder als Start-, Zwischen- oder Endereignis. In BPMN sind diese drei Verwendungen vorzufinden, allerdings als eigenständige Elemente mit verschiedenen Typen (OMG 2011, S. 261-262); beispielsweise Nachrichten- oder Timerereignisse. Der richtige Typ kann nach Vanderhaeghen et al. (2005, S. 88) durch definierte Ausnahmeklassen definiert werden, die prüfen, ob ein Ereignis ein Start- oder Endereignis ist.

Das von Hoyer et al. (2007, S. 193) aufgeführte Problem kann gelöst werden, wenn das Ereignis mit seinen Vorgängern und Nachfolgern bei der Transformation berücksichtigt wird. Dieser Ansatz ist bei allen Transformationsregeln von einem Quellmodell im ProSiT Konzept berücksichtigt. Eine semantisch richtige Zuordnung kann mit dem Ansatz von Decker et al. (2009, S. 101-102) erreicht werden. Die Bezeichnung eines EPK Ereignisses wird extrahiert und anhand des verwendeten Verbes der Typ bestimmt[14]. Zwei Beispiele werden von den Autoren aufgeführt. Das Startereignis „Kunde ruft an" wird zu einem BPMN Startereignis des Typs Nachricht. Beim Startereignis „Messekontakt wurde protokolliert" wird hingegen kein Typ für das Startereignis festgelegt. Um diesen Ansatz umzusetzen, wäre eine Datenbank mit Verben notwendig, welche mit Ereignistypen verknüpft sind. Dieses Konzept der Extraktion und Analyse von Bezeichnungen der Prozesselemente findet auch Anwendung im Rahmen der Normalisierung des ProSiT Konzepts[15].

Im zweiten Teilbereich des ProSiT Konzepts erfolgt die Normalisierung des konzeptionellen zum konsistenten Transformationsmodell. Die Normalisierung ist in drei Gruppen aufgeteilt: automatische und semiautomatische Normalisierungsregeln sowie die manuelle Normalisierung. Automatische Normalisierungsregeln erfolgen ohne Interaktion mit einem Benutzer, während semi-

14 In der eEPK haben Ereignisse eine Beschreibung, bei BPMN hingegen werden verschiedene Typen von Ereignissen über ein Symbol differenziert.

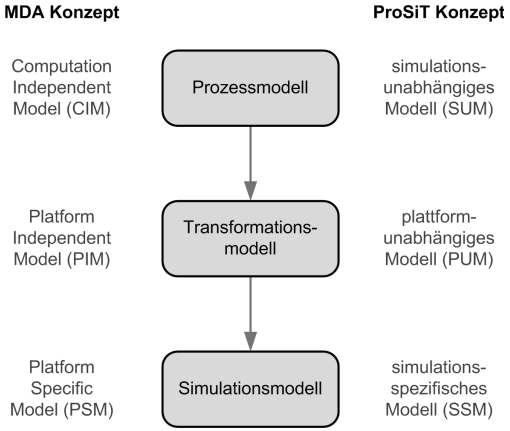
15 Die Gedanke für die Verwendung von linguistischen Regeln wurde unabhängig von Decker et al. (2009) entwickelt und ebenfalls im Jahr 2009 im Kontext des ProSiT Konzepts in Kloos et al. (2009) vorgestellt.

automatische Regeln eine Eingabe eines Benutzers benötigen. Bei diesen Normalisierungsregeln erfolgt die Extraktion und Analyse der Bezeichnungen der Prozesselemente. Diese Regeln werden als linguistische Normalisierungsregeln bezeichnet. Im Rahmen der manuellen Normalisierung kann das Modell durch den Benutzer, teils geführt, erweitert oder reduziert werden. Ist die Normalisierung abgeschlossen, dann liegt ein konsistentes Transformationsmodell vor.

Der dritte Teilbereich umfasst die Überführung des konsistenten Transformationsmodells in die Simulationsumgebungen. Zwei Regelgruppen liegen für diesen Teilbereich vor. Anwendbarkeitsregeln prüfen ob das Modell in die ausgewählte Simulationsumgebung überführt werden kann. Dies erfolgt in dem nach Konstrukten gesucht wird, die in der Simulationsumgebung nicht realisiert werden können. Ist das konsistente Transformationsmodell überführbar, werden Transformationsregeln angewendet. Diese berücksichtigen aber im Gegensatz zu den Transformationsregeln der Quellmodelle nicht die Verknüpfungsmöglichkeiten, da jedes Element direkt überführbar ist.

Durch dieses Vorgehen kann das Transformationskonzept auch aus Sicht der Model Driven Architecture (MDA) betrachtet werden. Dieses wurde in Kloos und Nissen (2010) thematisiert und wird an dieser Stelle kurz aufgeführt. Das MDA Konzept definiert drei Modelltypen: Computer Independent Model (CIM), Platfrom Independet Model (PIM) und Platform Specifc Model (PSM) (Gruhn et al. 2006, S. 120). Der Detaillierungsgrad der drei Modelle sowie die Zugehörigkeit zu einer Umsetzung oder Implementierung ist jeweils steigend. In einer Analogie zum ProSiT Konzept, kann ein Geschäftsprozessmodell als CIM angesehen werden und wird als simulationsunabhängiges Modell bezeichnet. Das PIM würde dem Transformationsmodell entsprechend. Das Modell ist für den Einsatz im Rahmen einer Simulationsumgebung vorgesehen, jedoch unabhängig von einer konkreten Umsetzung. In Kloos und Nissen (2010, S. 112) wird das Transformationsmodell daher als Plattform unabhängiges Modell aufgefasst. Das PSM hingegen entspricht der Umsetzung in einer Simulationsumgebung sowie aus

Modellsicht dem Simulationsmodell. Da eine direkte Umsetzung vorliegt, wird dieses als simulationsspezifisches Modell aufgeführt. Diese Analogie zwischen MDA und dem ProSiT Konzept ist in Abbildung 41 aufgeführt.



Kloos und Nissen (2010, S. 120)

Abbildung 41: Analogie zwischen MDA und dem ProSiT Konzept

Die Überführung des Prozessmodells in das Transformationsmodell sowie des Transformationsmodells in das Simulationsmodell erfolgt über Transformationsregeln. Gemeinsam mit den Normalisierungsregeln bilden diese in ihrer Gesamtheit die Regelbasis. In Tabelle 26 ist die Regelbasis hinsichtlich hinsichtlich ihres Zweckes aus Sicht der MDA klassifiziert.

Tabelle 26: Klassifikation der Transformationsregeln

		Verfeinerung	Abstraktion	Migration	Refaktorierung	Optimierung	Darstellungsform
Prozessmodell							
Syntaxregel	(SR)						
Vorbereitungsregel	(PR)		X				
Transformationsregel	(TR)			X			(X)
Transformationsmodell							
Automatische Normalisierungsregel	(aNR)		X		X		
Semiautomatische Normalisierungsregel	(sNR)	X	X				
Manuelle Normalisierung	(mNR)	X	X		X		
Simulationsmodell							
Anwendbarkeitsregel	(AR)						
Transformationsregel	(TR)			X			(X)

Nach Kloos und Nissen (2010, S. 114)

Die sechs Spalten mit dem Regelzweck richten sich nach den Ausführungen von Gruhn et al. (2006, S. 151-152). Eine Anreicherung des Models um zusätzliche Information erfolgt durch den Einsatz einer Regel, welche die Verfeinerung als Schwerpunkt hat. Dies trifft bei der Regelbasis nur auf semiautomatische Normalisierungsregeln sowie die manuelle Normalisierung zu. Die Abstraktion ist das Gegenstück zu dieser Regelart, in dem Informationen aus dem Modell entfernt werden. Im Kontext des ProSiT Konzepts trifft dies auf Informationen zu, welche nicht für die Simulation notwendig sind. Eine Migration liegt nach Gruhn et al. (2006, S. 151) vor, wenn die technologische Plattform für das Modell geändert wird. Gemäß der Analogie zur MDA trifft dies auf die Überführung zwischen den einzelnen Modellen zu; realisiert durch Transformationsregeln. Ändert sich die innere Struktur eines

Modells, ohne dass sich das äußere Verhalten ändert, dann entspricht dies der Refakturierung. Eine Refakturierung ist nur im Rahmen der Normalisierung vorzufinden. Hierbei werden beispielsweise automatische Normalisierungsregeln angewendet, um eine konzeptuelle Aktivität mit Markern in eine konsistente Aktivität umzuwandeln. Ebenfalls kann durch manuellen Eingriff in das Transformationsmodell die innere Struktur verändert werden. Als eine Optimierung bezeichnen Gruhn et al. (2006, S. 156) die Optimierung des Modells. So soll weniger Speicherplatz oder ein geringerer Rechenzeitbedarf als Ergebnis der Optimierung stehen. Eine derartige Regelart ist im ProSiT Konzept nicht vorgesehen. Regeln, die den Zweck verfolgen die Darstellungsform zu ändern, ohne dabei die innere Struktur oder das äußere Verhalten zu verändern ist der sechste Zweck einer Regel. Dies trifft auf die Transformationsregeln zu, jedoch ist dies nicht der Hauptzweck der Regeln, der bei der Migration eingeordnet wird. In Tabelle 26 wird daher eine Klammer um die Markierung gesetzt, um dieses Sachverhalt auszudrücken.

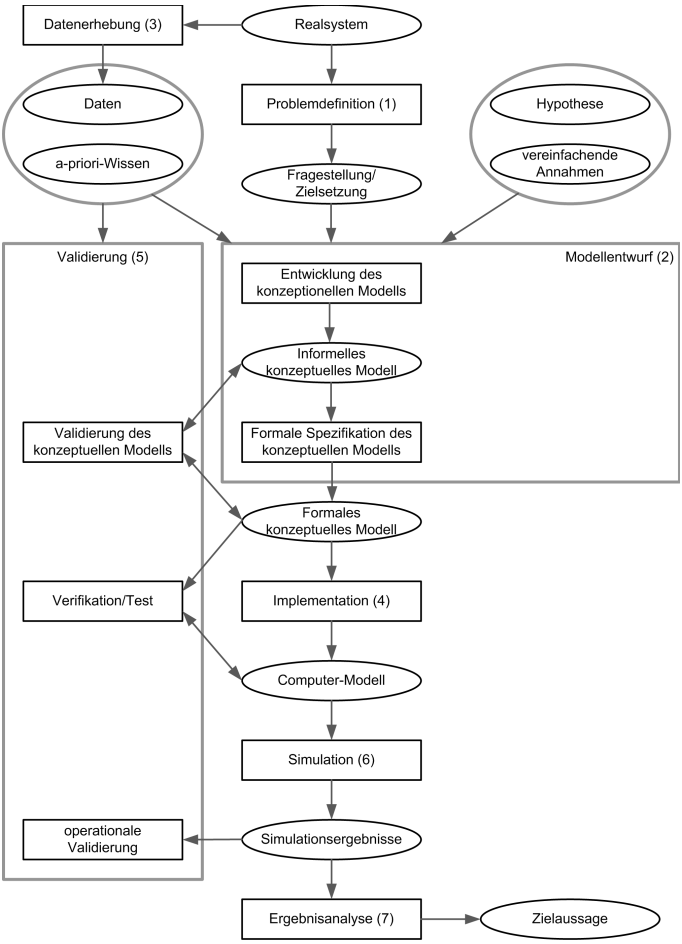
Lediglich bei der Normalisierung ist mehr als ein Zweck aufgeführt. Semiautomatische Normalisierungsregeln können entweder das Modell verfeinern oder abstrahieren. Zu beachten ist die Sprache, in der die Regeln formuliert werden. Ist eine Regel in englischer Sprache formuliert, ist diese unabhängig von der verwendeten Sprache, mit denen das semiformale Modell beschrieben wird. Eine Regel in deutscher Sprache kann hingegen nur für deutsch formulierte Prozessmodelle verwendet werden. Eine Überführung in andere Sprache ist für die Regel jeweils zu prüfen.

3.3 Einordnung des ProSiT Konzepts in Vorgehensmodelle

Für die Anwendung des ProSiT Konzepts, im Rahmen einer Simulationsstudie, wird kein eigenes Vorgehensmodell definiert. Stattdessen wird das Konzept in bestehende Vorgehensmodelle eingeordnet. Diese Vorgehensmodelle adressieren entweder die Simulation von Geschäftsprozessen oder sind allgemein für die Durchführung einer Simulationsstudie konzipiert.

Fünf Vorgehensmodelle werden betrachtet. Historisch wurde in der ersten Konzeption das Vorgehensmodell von Page berücksichtigt (Himmelreich 2007, S. 38-39), wodurch auf dieses etwas ausführlicher eingegangen wird. Aufgrund der verwendeten Definition des Begriffs der VDI für die Simulation wird das entsprechende Vorgehensmodell der VDI betrachtet. Neben diesen allgemeinen Vorgehensmodellen werden noch drei aus dem Kontext der Simulation von Geschäftsprozessen betrachtet (Allweyer 2004; Neumann et al. 2005; Gadatsch 2010).

Das Vorgehensmodell von Page (1991, S. 12-18) ist in Abbildung 42 dargestellt. Sieben Schritte stehen in diesem im Vordergrund. Bei der Problemdefinition (1) sind die Ziele der Simulationsstudie festzulegen und die zu untersuchende Fragestellung. Aufbauend auf dieser wird bei dem Modellentwurf (2) zunächst eine Abgrenzung des betrachteten Systems vollzogen und ein informales konzeptuelles Modell entwickelt. Im Kontext von Prozessen wäre dies der Ablauf. Die Formalisierung des konzeptuellen Modells ist der nächste Schritt des Modellentwurfs. Hierfür werden die Elemente des Systems mit entsprechenden Daten angereichert, damit das Modell in das mathematische Modell eines Simulationsmodells überführt werden kann.



Page (1991, S. 12)

Abbildung 42: Vorgehensmodell zur Durchführung einer Simulationsstudie nach Page

Beim Modellentwurf wird festgelegt, welche Daten das Modell benötigt. Die Ergebung der notwendigen Daten erfolgt im Schritt der Datenerhebung (3) parallel zum Modellentwurf. Die Daten bilden die Grundlage für die Erstellung des formalen konzeptionellen Modells. Dass durch diese beiden Schritte erstellte Modell, wird im nächsten Schritt, der Implementation (4), in ein ausführbares Computermodell

überführt. Dieses kann entweder mit einer simulierbaren Programmiersprache oder einer grafischen Simulationsumgebung realisiert werden. Das Computermodell entspricht einem digital ausführbarem konzeptuellen Modell (Page 1991, S. 15-16).

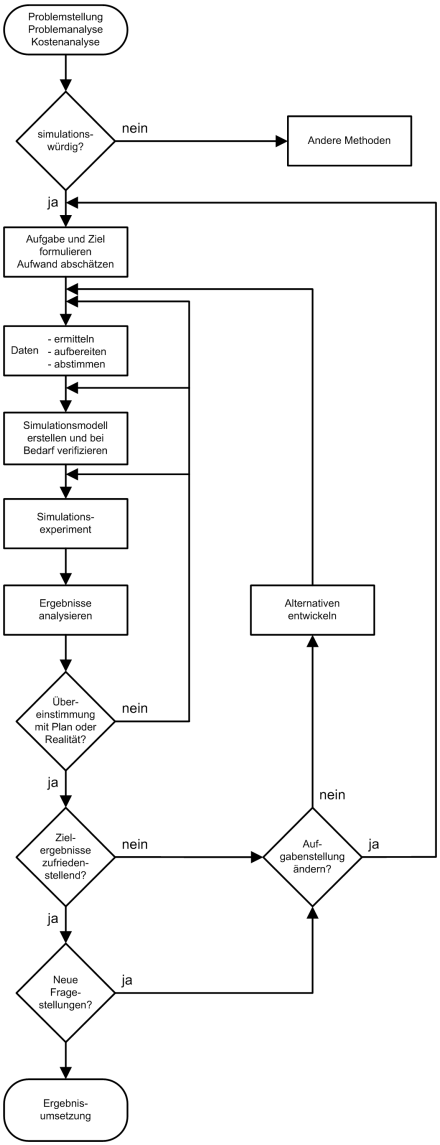
Vor dem eigentlichen Simulationslauf muss das erstellte Simulationsmodell hinsichtlich seiner Gültigkeit geprüft werden. Die Validierung (5) hat anhand des Realsystems und der daraus abgeleiteten Fragestellung zu erfolgen. Zunächst ist das konzeptuelle Modell zu validieren, ob dieses das Realsystem hinreichend beschreibt. Das Computermodell ist bezüglich seiner korrekten Abbildung des konzeptuellen Modells zu verifizieren. Nach dem eigentlichen Simulationslauf ist eine abschließende operationale Modellvalidierung durchzuführen, ob der Ablauf das Computermodell mit der Realität übereinstimmt. Ein Beispiel hierfür ist ein Vergleich der Ergebnisse des Simulationslaufs mit den einer Periode vorliegenden Ergebnissen der Realität, wenn diese zur Verfügung stehen. Wird bei einer dieser drei Prüfungen ein Mangel festgestellt, müssen entsprechend des Ergebnisses die vorhergehenden Phasen wiederholt werden (Page 1991, S. 16-17).

Das verifizierte Computermodell bildet das Fundament für die eigentliche Simulation (6). Für jeden Simulationslauf werden die Parameter leicht geändert und die Ergebnisse für die anschließende Ergebnisanalyse (7) aufbereitet. Bei dieser Phase werden die Ergebnisse bewertet, um Aussagen auf das Realsystem zu treffen und um die definierte Fragestellung zu beantworten. Einschränkungen, die während der vorherigen Phasen vorgenommen wurden, müssen bei der Ergebnisanalyse berücksichtigt werden, damit die Ergebnisse im Angesicht des Realsystems richtig interpretiert werden (Page 1991, S. 17-18). Eine zusätzliche achte Phase ist die parallel zum Modellbildungsprozess laufende Dokumentation der einzelnen Schritte. Dies umfasst neben der Problemstellung eine Beschreibung der einzelnen Modelle, eine Dokumentation der Validierung und die

Darstellung der unterstellten Annahmen bei den einzelnen Simulationsläufen (Page 1991, S. 18).

Aus diesem Vorgehensmodell wird Schritt 2 bis 4 unterstützt. Beim Modellentwurf liegen Geschäftsprozessmodelle als informelle konzeptuelle Modelle vor. Die Unterstützung erfolgt für die formale Spezifikation des konzeptuellen Modells, mittels der Transformationsregeln zum Transformationsmodell sowie den Normalisierungsregeln. Die Datenerhebung wird ebenfalls durch Normalisierungsregeln sowie der Tätigkeitssicht unterstützt. Die Implementierung in ein ausführbares Computermodell hingegen erfolgt wieder über Transformationsregeln zu den Simulationsumgebungen.

Wird das ProSiT Konzept in das Vorgehensmodell der VDI (1995b, S. 11), in Abbildung 43, eingeordnet, dann unterstützt dieses die Schritte: Datenermittlung, -aufbereitung und -abstimmung sowie den Schritt „Simulationsmodell erstellen“. Die Zuordnung des ProSiT Konzepts kann hierbei analog zu den Ausführungen zum Vorgehensmodell von Page erfolgen. Auf die Simulierwürdigkeit wurde in Kapitel 2.4 eingegangen. Diese ist aber nicht Teil des ProSiT Konzepts.



Nach VDI (1995b, S. 11)

Abbildung 43: Vorgehensmodell nach VDI 3633

Aus dem Umfeld der Simulation von Geschäftsprozessen wird zunächst das Vorgehensmodell von Allweyer (2005, S. 206) betrachtet. Dieser definiert sechs Schritte um eine Simulationsstudie durchzuführen:

1. Zielsetzung festlegen

2. Informationen beschaffen

3. Modellierung

4. Validierung

5. Simulationsexperimente durchführen

6. Ergebnisse analysieren und bewerten

Das in Gadatsch (2010, S. 226-227) aufgeführte Vorgehensmodell entspricht bis auf einem Schritt dem Vorgehensmodell von Allweyer. Der Schritt Modellierung wird in die zwei Schritte – Modellbildungsprozess und Implementierung – unterteilt. Nach Gadatsch (2010, S. 226-227) sind daher sieben Schritte zur Durchführung einer Simulationsstudie notwendig.

Während bei Allweyer das ProSiT Konzept nur den Schritten zwei und drei zuzuordnen ist, werden daher bei Gadatsch die Schritte zwei bis vier unterstützt. Unter dem Modellbildungsprozess soll hierbei nicht die Modellierung von Geschäftsprozessen verstanden werden, sondern die Erzeugung des Transformationsmodells aus den Geschäftsprozessmodellen sowie dessen Normalisierung.

Neumann et al. (2005, S. 438-440) unterteilen die Durchführung einer Simulationsstudie in neuen Phasen. Bei jeder Phase kann es zu einem Rücksprung zu einer vorhergehenden Phase kommen, die Abfolge ist daher nicht sequenziell aufzufassen:

- Planung
- Analyse
- Datendefinition und -erhebung
- Konstruktion
- Simulationsdurchführung
- Modellüberprüfung
- Ergebnisinterpretation
- Berechnungsexperimente
- Ergebnisdarstellung

Im Vergleich zu Allweyer und Gadatsch wird die Ergebnisanalyse und -bewertung in drei Schritte unterteilt. Vom ProSit Konzept werden die Phasen der Datenerhebung sowie die Konstruktion des Simulationsmodells unterstützt.

Durch die Einordnung in verschiedene Vorgehensmodelle wird die These aufgestellt, dass der Ansatz unabhängig von einem konkreten Vorgehensmodell einsetzbar ist.

3.4 Überführung der Quellmodelle in das Transformationsmodell

In den Unterkapiteln werden ausgewählte Transformationsregeln einer Prozessmodellierungsnotation in das ProSiT Konzept erläutert. Die vollständige Regelbasis ist im Anhang A.1 aufgeführt. Jede Regel beginnt mit ihrer sprachlichen Formulierung. Anschließend folgt für die ausgewählten Transformationsregeln die Herleitung und abschließend eine grafische Darstellung anhand eines abstrakten Beispiels.

3.4.1 Von der eEPK zum Transformationsmodell

Die Transformationsregeln leiten sich aus den Verknüpfungsmöglichkeiten der EPK ab (Abbildung 2). Die von Mendling und Nüttgens (2003a, S. 22) vorgestellt Möglichkeiten stellen alle Kombinationenmöglichkeiten des Kontrollflusses der EPK dar. Zusätzliche Elemente der eEPK werden lediglich noch zusätzlich mit einer Funktion verknüpft.

Syntaxregeln – eSR (EPC Syntax Rule)
Zur Sicherstellung der syntaktischen Validierung der EPK sieht das ProSiT Konzept die Anwendung von Syntaxregeln vor (Kloos et al. 2011, S. 25). Zur syntaktischen Prüfung wird auf die Arbeit von Mendling und Nüttgens (2003a, S. 21-24) verwiesen. Die Autoren vollzogen eine formale Spezifikation, nach der eine EPK syntaktisch korrekt ist. Nachfolgend ist die Spezifikation für Zwischenereignisse dargestellt (Mendling und Nüttgens 2003a, S. 23). Diese besagt, dass ein Zwischenereignis einen Vorgänger und einen Nachfolger haben muss.

$$\textit{Inner Events}: \forall v \in E_I: \ |\rightarrow v| = 1 \wedge |v \rightarrow| = 1.$$

Auf diese Spezifikationen kann aufgrund der EPC Markup Language zurückgegriffen werden, die für die eEPK als Grundlage verwendet wird. Syntaktische Regeln werden nicht explizit aufgeführt. Für diese wird auf Mendling und Nüttgens (2003a, S. 21-24) sowie Mendling und Nüttgens (2003b, S. 133-138) verwiesen.

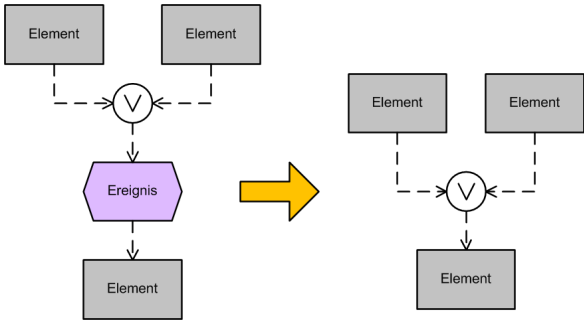
Vorbereitungsregeln – ePR (EPC Preparation Rule)

Die Vorbereitungsregeln der eEPK sind hauptsächlich dafür konzipiert, Ereignisse zu entfernen, welche für eine Simulation keinen Mehrwert erzeugen. Als ein Beispiel hierfür wird die Vorbereitungsregeln ePR_3 aufgeführt. Alle nicht entfernten Ereignisse werden durch Transformationsregeln in andere Elemente überführt. Eine besondere Vorbereitungsregel ist ePR_6. Diese fasst zwei aufeinanderfolgende verzeigende exklusive Operatoren zusammen.

Vorbereitungsregel ePR_3
If an event has an OR operator as its predecessor and it has a successor and the OR operator has more than one predecessor, then it is removed.

Die Vorbereitungsregel ePR_3 identifiziert ein Ereignis, welches einen Nachfolger und einen OR Operator als Vorgänger hat, der mehr als einen Vorgänger hat. Beim OR Operator handelt es sich um einen zusammenführenden Operator. Das Ereignis wird bei jedem Prozessdurchlauf erreicht und kann als sequenzielles Zwischenereignis aufgefasst werden. In der Literatur wird diese Art von Ereignissen auch als trivial Ereignisse (Allweyer 2005, S. 184; Becker et al. 2009, S. 53) bezeichnet. Um die Lesbarkeit der Prozessmodelle zu erhöhen, werden trivial Ereignisse meist entfernt oder nicht modelliert. Für die Durchführung einer Simulationsstudie bieten diese Zwischenereignisse keinen informationellen Mehrwert. Daher können diese aus der Betrachtung entfernt werden.

Darstellung der Vorbereitungsregel

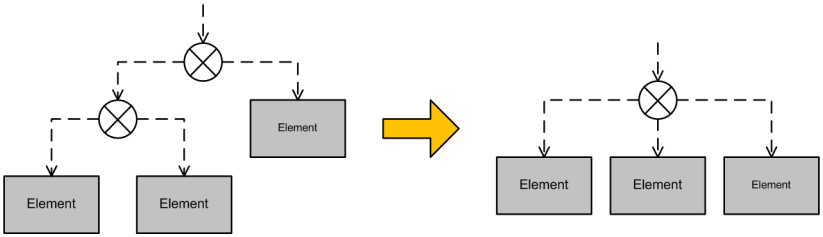


Vorbereitungsregel ePR_5
If a XOR operator has more than one successors and a XOR operator, with more than one successors, as its predecessor, the XOR Operator is removed and all successors will become successors of its predecessor XOR.

Die Vorbereitungsregel ePR_5 identifiziert einen XOR Operator, der mehr als einen Nachfolger hat und einen XOR Operator als Vorgänger, der ebenfalls mehr als einen Nachfolger hat. Dieses Konstrukt stellt zwei aufeinanderfolgende exklusive Entscheidungen dar[16]. Ein XOR Operator wird im ProSiT Ablaufdiagramm als wahrscheinlichkeitsbasierte Entscheidung interpretiert (siehe eTR_9 und eTR_{15}). Würden beide XOR Operatoren in das ProSiT Ablaufdiagramm überführt, dann müsste zwischen beiden Operatoren ein exklusiver Schlüssel eingefügt werden. Aus semantischer Sicht können diese einzelnen Operatoren als ein Operator zusammengefasst werden. Als Folge der Vorbereitungsregel muss im Rahmen der Normalisierung auf die entsprechende Wahrscheinlichkeit geachtet werden.

16 Dieser Fall wird in Abbildung 2 auf Seite 28 in der Zeile XOR_{FES} und Spalte XOR_{FES} beschrieben.

Darstellung der Vorbereitungsregel



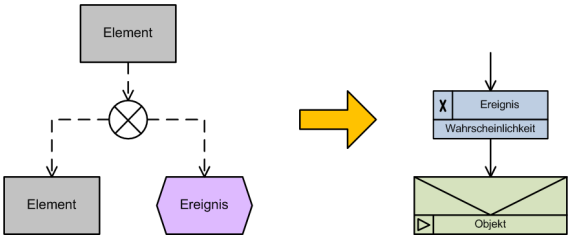
Transformationsregeln – eTR (EPC Transformation Rules)

Die Transformationsregeln überführen die durch die Vorbereitungsregeln präparierte eEPK in das ProSiT Ablaufdiagramm. Für eine syntaktisch vollständige Überführung sind 19 Transformationsregeln notwendig. An dieser Stelle werden 5 Transformationsregeln beschrieben, die besondere Aspekte der eEPK hervorheben.

Transformationsregel eTR_7
If an event has a XOR operator as its predecessor and it has not a successor and the XOR operator has one predecessor, then it will be an exclusive key and a sink, that is the successor of the key.

Die Transformationsregel eTR_7 identifiziert ein Ereignis mit einen XOR Operator als Vorgänger und keinen Nachfolger. Hat ein Ereignis keinen Nachfolger, ist dieses ein Endereignis. Ein XOR Operator, der nur einen Vorgänger hat, ist ein trennender Operator. Bei einem trennenden XOR Operator werden Prozesspfade nur ausgeführt, wenn ein nachfolgendes Ereignis eintritt. Diese bedingte Ausführung muss bei der Transformation berücksichtigt werden. Das Endereignis wird das Endereignis in einen exklusiven Schlüssel und eine Senke als Nachfolger des Schlüssels umgewandelt.

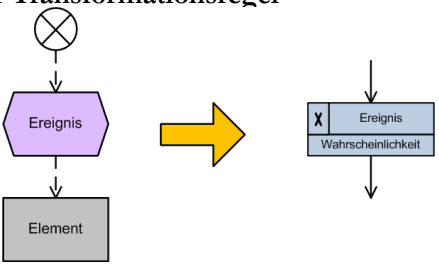
Darstellung der Transformationsregel



Transformationsregel eTR_9
If an event has a XOR operator as its predecessor and it has a successor, then it will be an exclusive key.

Ein Ereignis mit einem XOR Operator als Vorgänger und einem Nachfolger wird durch die Transformationsregeln eTR_9 identifiziert. Aufgrund der Vorbereitungsregel ePR_3 wurden alle Ereignisse entfernt, die auf einen zusammenführenden XOR Operator folgen und einen Nachfolger haben. Das Ereignis muss daher auf einen verzweigenden Operator folgen. Aufgrund des Nachfolgers beschreibt das Zwischenereignis die Bedingung, nach der ein Prozesspfad auszuführen ist. Diese Bedingung wird im ProSiT Ablaufdiagramm in Form eines wahrscheinlichkeitsbasierten exklusiven Schlüssels abgebildet.

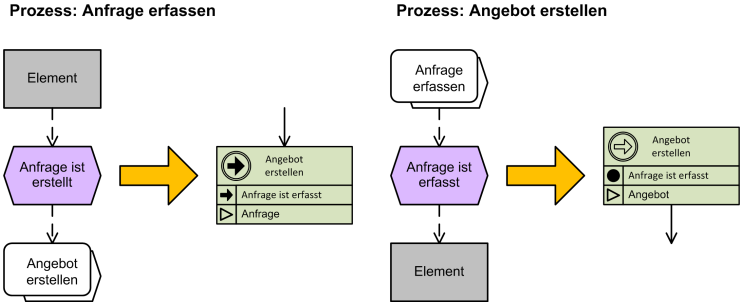
Darstellung der Transformationsregel



Transformationsregel eTR_{10}
If an event has a process interface as its predecessor and it has a successor, then the event and the process interface will be a target point.

Die Transformationsregel eTR_{10} identifiziert ein Ereignis mit einer Prozessschnittstelle als Vorgänger und einen Nachfolger. Das Ereignis stellt ein Startereignis für den Prozess dar; die Prozessschnittstelle einen Verweis auf einen vorhergehenden Prozess auf gleicher Hierarchiestufe. Aufseiten des vorhergehenden Prozesses gibt es eine Prozessschnittstelle mit einem Ereignis als Vorgänger, die auf den betrachteten Prozess verweist. Die Verknüpfung zwischen den beiden Prozessen erfolgt über gleich benannte Ereignisse. Da es sich um einen einseitigen Verweis[17] handelt, können mehrere vorgelagerte Prozesse auf einen Zielprozess verweisen. Das Ereignis und die vorgelagerte Prozessschnittstelle werden in einen Zielpunkt überführt.

Anwendungsbeispiel



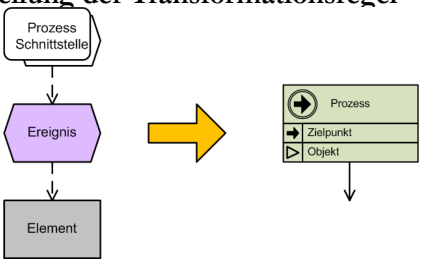
Im Gegensatz zur Transformationsregel eTR_1 stellt das Ereignis keine Quelle dar, mit der Marken erzeugt werden. Marken werden von vorhergehenden Prozessen an den Zielpunkt übergeben. Drei Daten enthält der Zielpunkt: den Namen des Prozesses, die Bezeichnung des Ereignisses und das Prozessobjekt. Um einen eindeutigen Verweis zu erstellen, erhält die Sprungpunkte (eTR_{11}) den Namen aus der

17 Aus Sicht eines Simulationslaufes ist ein Rücksprung zu den Quellprozessen nicht notwendig. Marken bewegen sich nur in eine Richtung, von der Quelle in Richtung Senke.

Prozessschnittstelle, den Text des Ereignisses sowie das Prozessobjekt des eigenen Prozesses.

Mittels Ziel- und Sprungpunkt kann ein Wechsel des Objekts realisiert werden, wie dies im obigen Beispiel dargestellt ist. Im Rahmen der Normalisierung ist aber zu Prüfung, ob jeder Sprungpunkt auf einen Zielpunkt verweist.

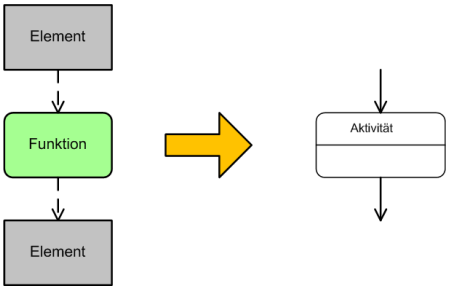
Grafische Darstellung der Transformationsregel



Transformationsregel eTR_{12}
If a function has not a linked process, then it will be an activity. If participants are related to the function, they will be added as a human resource to the activity.

Eine Funktion, die auf keinen weiteren Prozess verweist, wird von Transformationsregeln eTR_{12} verarbeitet. Nach Keller et al. (1991, S. 11) beschreibt eine Funktion „[...] die Durchführung eines betrieblichen Vorgangs, der zur Erfüllung eines Unternehmensziels beiträgt." Aus dem Begriff der Durchführung leitet sich eine Zeitspanne ab. Um einen betrieblichen Vorgang durchzuführen, sind Ressourcen notwendig. Bei einer eEPK werden diese Ressourcen der Funktion angehängt, beispielsweise eine Stelle. Zeitverbrauchende Vorgänge die Ressourcen beanspruchen, werden im ProSiT Ablaufdiagramm als Aktivitäten umgesetzt.

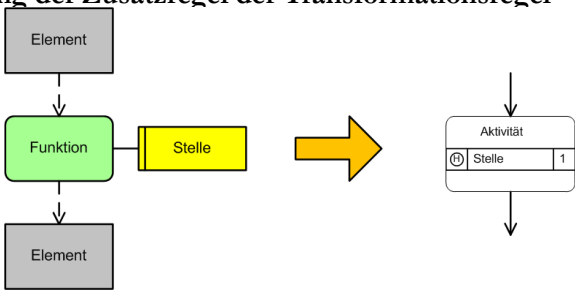
Darstellung der Transformationsregel



Zusatzregel bei der Transformation von Funktionen

Neben dieser Transformation hat die Transformationsregel eTR_{12} eine Zusatzregel. Wenn an die Funktion eine Stelle angefügt ist, dann wird diese bei der Aktivität als menschliche Ressource aufgenommen.

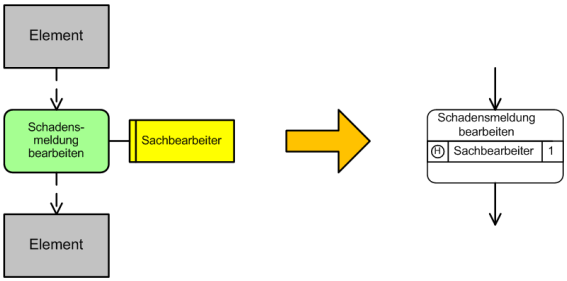
Darstellung der Zusatzregel der Transformationsregel



Wenn an eine Funktion eine Organisationseinheit in Form einer Stelle angefügt ist – im EPML Format das Element Participant –, dann wird die Stelle der Aktivität als menschliche Ressource hinzugefügt.

Anwendungsbeispiel

Neben dieser schematischen Darstellung ist nachfolgend ein Anwendungsbeispiel dargestellt. Das Funktionsbeispiel ist aus Allweyer (2005, S. 244) entnommen.



Wie aus der Formulierung der Transformationsregel hervorgeht, werden lediglich Personen beziehungsweise Stellen in das ProSiT Ablaufdiagramm überführt. Das zugrunde gelegte EPML Format definiert als weitere Ressource lediglich ein Informationssystem (Application). Stellen und Informationssysteme sind die beiden Elemente, die bei einer eEPK modelliert werden. Beispiele hierfür sind bei Becker et al. (2009, S. 52-53) oder Gadatsch (2010, S. 190) zu finden.

Im Rahmen des ProSiT Konzept erfolgt keine Simulation der Verfügbarkeit von Informationssystemen. Es wird unterstellt, Informationssysteme sind immer verfügbar. Für diese ist daher keine Zusatzregel definiert. Soll ein Informationssystem in der Simulation von Geschäftsprozessen berücksichtigt werden, wird auf den Artikel von Schmietendorf und End (2009) verwiesen.

Bei einer eEPK können weitere Ressourcen einer Funktion angehängt werden. Sind weitere Ressourcen zu berücksichtigen, muss für diese eine weitere Zusatzregel definiert werden. Ein Beispiel könnte ein Raum sein: „If a room is related to the function, it will be added as a stationary resource to the activity." Eine Zusatzregel ist nach dem folgenden Schema zu formulieren:

If a [p] is related to the function, it will be added as a [q] resource to the activity.

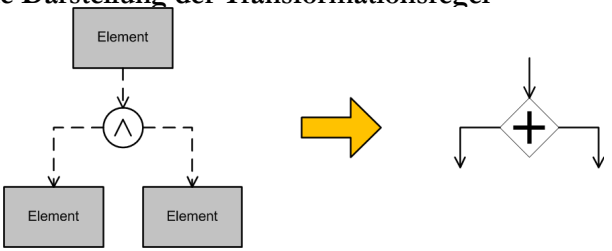
Während [p] die Ressource darstellt, definiert [q] den Typ der Ressource. Die Einordnung der Ressource in eine der vier Typen ist optional und kann ebenfalls bei der Normalisierung erfolgen. Das Muster unterscheidet darüber hinaus, ob es eine Ressource nur einmal einer Funktion (Raum) oder mehrere Ressourcen zugewiesen werden können (Participant). Dieser Aspekt wird mittel Singular und Plural in der Regel unterschieden.

Neben Ressourcen werden im EPML Format auch Daten in Form von Input und Output an eine Funktion angeheftet. Daten werden aber in das ProSiT Konzept nicht überführt; es gilt die Annahme, dass alle Daten zum Ausführen einer Aktivität vorhanden sind.

Transformationsregel eTR_{14}
If an AND operator has one predecessor and more than one successor, then it will be a parallel split gateway and if the predecessor is an OR operator that has more than one successor, it will get an inclusive key as predecessor of the gateway and if the predecessor is a XOR operator that has more than one successor, it will get an exclusive key as predecessor of the gateway.

Die Transformationsregel eTR_{14} identifiziert einen verzweigenden AND Operator. Bei einem verzweigenden AND Operator werden alle nachfolgenden Prozesspfade ausgeführt. Dies entspricht dem parallelen Gateway im ProSiT Ablaufdiagramm.

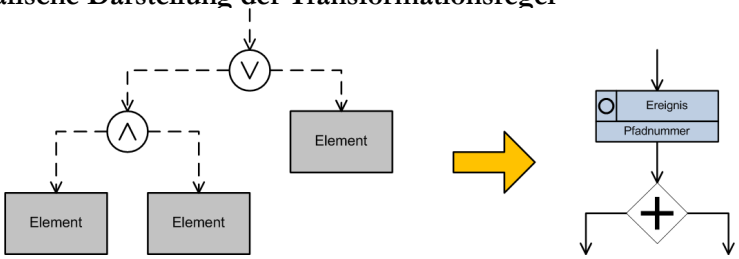
Grafische Darstellung der Transformationsregel



Zusatzbedingung: Verzweigender OR Operator als Vorgänger

Neben dieser Hauptbedingung sind zwei Zusatzbedingungen definiert. Die Erste prüft den Vorgänger nach einem verzweigenden OR Operator. Liegt dieser vor, müssen die Nachfolger des AND Operators Ereignisse sein. Diese Ereignisse müssen eintreten, damit der Prozesspfad mit dem AND Operator ausgeführt wird. Durch die Vorbereitungsregel ePR_2 wurden alle Ereignisse entfernt, die einen AND Operator als Vorgänger haben. Damit das resultierende ProSiT Ablaufdiagramm fehlerfrei ist, muss für den Prozesspfad mit dem AND Operator eine Wahrscheinlichkeit angegeben werden. Dies erfolgt mit einem inklusiven Schlüssel, der dem AND Operator als Vorgänger dient. Da der Schlüssel nicht von einem Ereignis abgeleitet wird, muss dieser im Rahmen der Normalisierung mit einem sprechenden Ereignistext versehen werden.

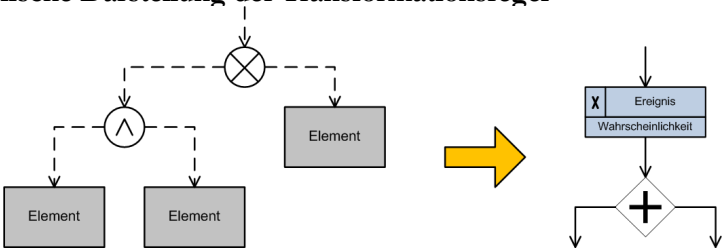
Grafische Darstellung der Transformationsregel



Zusatzbedingung: Verzweigender XOR Operator als Vorgänger

Analog zu den Ausführungen bei der Zusatzbedingung zum verzweigenden OR Operator, gilt die gleiche Argumentation für die zweite Zusatzbedingung; einem XOR Operator als Vorgänger. Anstelle eines inklusiven Schlüssels wird aber ein exklusiver Schlüssel eingefügt.

Grafische Darstellung der Transformationsregel



3.4.2 Von der BPMN zum Transformationsmodell

In diesem Kapitel wird die Regelbasis erläutert, um einen BPMN Prozess in das ProSiT Ablaufdiagramm zu überführen. Wenn nicht anders angegeben wird davon ausgegangen, dass die Nachfolger mit einem Sequenzfluss verknüpft sind.

Syntaxregeln – bSR (BPMN Syntax Rule)
Für die syntaktischen Regeln um einen Prozess zu modellieren, wird auf die BPMN Spezifikationen verwiesen (OMG 2011, S. 145-314). Eine Aufführung der Regel erfolgt nicht. An ein BPMN Modell werden jedoch Bedingungen gestellt, um dieses in das ProSiT Ablaufdiagramm zu überführt.

Wie bei den Erläuterungen zur BPMN Notation aufgeführt, wird der Ereignis-Teilprozess im ProSiT Konzepts nicht berücksichtigt. Die Syntaxregel bSR_1 schließt demnach die BPMN Prozesse aus, die einen Ereignisteilprozess beinhalten.

Syntaxregel bSR_1
If the process contains an event sub process, than the model can not be converted to the ProSiT sequence diagram.

Eine weitere Syntaxregel ist für Teilprozesse und die Transaktionen definiert. Diese können angeheftete Ereignisse haben (OMG 2011, S. 178-179). Tritt eines dieser Ereignisse ein, wird der Teilprozess abgebrochen und Transaktionen darüber hinaus kompensiert. Da bei den betrachteten Simulationsumgebungen keine Verknüpfung zwischen der Ausführung von zwei Aktivitäten vorherrscht, können parallel laufende Aktivitäten nicht abgebrochen werden, wenn ein angeheftetes Ereignis eintritt. Damit das Transformationsmodell in den Simulationsumgebungen semantisch korrekt umgesetzt werden kann, wird dieser Fall durch die Syntaxregel bSR_2 ausgeschlossen.

Syntaxregel bSR_2
If a sub process or a transaction have a parallel, inclusive or complex gateway and a boundary event, than the model can not be converted to the ProSiT sequence diagram.

Vorbereitungsregeln – bPR (BPMN Preparation Rule)

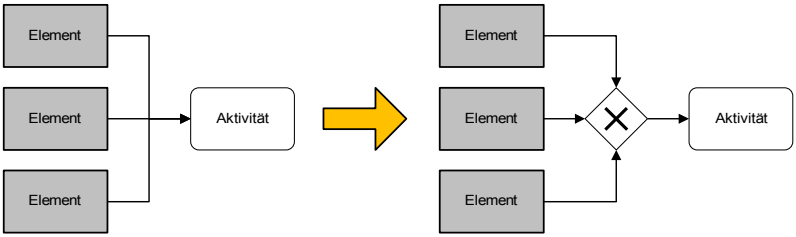
Die Vorbereitungsregeln für BPMN Prozesse bauen auf einem zweistufigen Konzept auf, um die Anzahl notwendigen Vorbereitungsregeln zu reduzieren. Die erste Serie der Vorbereitungsregeln bPR_{1_x} fokussiert auf, das zentrale Element des BPMN Prozesses, die Aktivität. Sie bereitet die Vorgänger- und Nachfolgerbeziehungen sowie Nachrichtenflüsse vor. Dadurch werden Spezialfälle umgangen, die separat als Vorbereitungsregel betrachtet werden müssten. Im Gegensatz zu den Vorbereitungsregeln der eEPK wird eine Regel auf mehrere Elementtypen angewendet. Die zweite Serie ePR_{2_x} baut auf den Ergebnissen der ersten Serie auf und bildet die Grundlage für die Transformationsregeln. Durch die Nachrichtenflüsse ist es notwendig, zwei Elemente vorzubereiten. Regeln der zweiten Serie rufen dafür ePR_{N_x} Regeln auf, die das Gegenstück des Nachrichtenflusses anpassen.

Vorbereitungsregelserie bPR_{1_x}

Vorbereitungsregel bPR_{1_3}
If an activity has more than one predecessor, then the activity will get an exclusive gateway as predecessor and all predecessors of the activity will get the predecessors of the gateway.

Die Vorbereitungsregel bPR_{1_3} identifiziert eine Aktivität mit mehr als einem Vorgänger. In diesem Falle muss nur eine Marke eines Vorgängers die Aktivität erreichen, damit diese ausgeführt werden kann (Allweyer 2009, 42). Zwischen den Vorgängern und der Aktivität wird daher ein zusammenführendes exklusives Gateway eingefügt.

Grafische Darstellung der Vorbereitungsregel

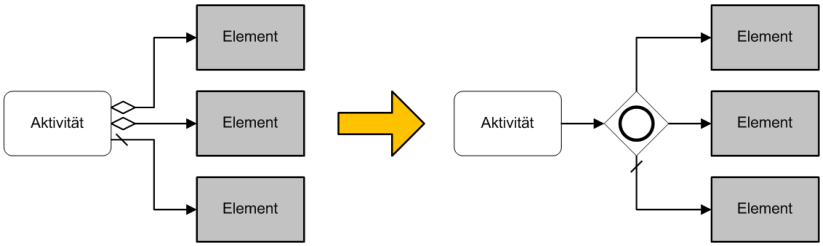


Vorbereitungsregel bPR_{1_5}
If an activity has one or more conditional flow or a default flow for the connection to its successors, then the conditional flows are replaced by a sequence flow and the activity will get an inclusive gateway as successor and all successors of the activity will become the successors of the gateway.

Neben Gateways kann eine Entscheidung eines ausführenden Prozesspfades auch über bedingte Flüsse erfolgen. Die Vorbereitungsregel bPR_{1_5} prüft, ob die Nachfolger mit einem bedingten Fluss verknüpft sind und ob ein Standardfluss verwendet wird. In der BPMN Notation stellt bedingte Flüsse ein logisches Oder dar. Abhängig von den verwendeten Bedingungen handelt es sich entweder um ein inklusives oder um ein exklusives Oder (Allweyer 2009, S. 38-39). Da im Rahmen des ProSiT Konzepts keine Prüfung der Bedingungen erfolgt, wird davon ausgegangen, dass es sich um ein inklusives Oder handelt. Dieses Wahl führt zwar nicht zu einem logischen Widerspruch, jedoch ist im Rahmen der Normalisierung zu prüfen, ob das inklusive Gateway durch ein exklusives Gateway zu ersetzen ist. Wäre jedoch ein Mechanismus vorhanden, die Bedingungen hinsichtlich des sich gegenseitigen Ausschließens automatisch zu überprüfen, dann kann die Vorbereitungsregel erweitert werden, wodurch entweder ein inklusives oder ein exklusives Gateway eingesetzt wird.

Die Vorbereitungsregel wandelt bedingte Flüsse in Sequenzflüsse um und fügt ein inklusives Gateway als Nachfolger der Aktivität ein. Falls ein Standardfluss verwendet wird, so wird dieser nicht von der Vorbereitungsregel verändert.

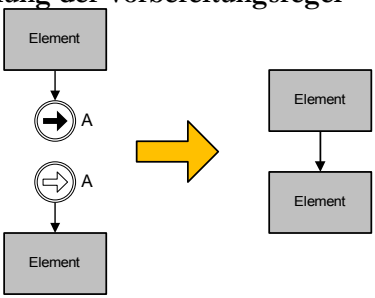
Grafische Darstellung der Vorbereitungsregel



Vorbereitungsregel bPR_{1_6}
If an intermediate link throwing event has the same labeling as an intermediate link catching event, then both elements are removed and the predecessor of the intermediate link throwing event is connected to the successor of the intermediate link catching event.

Die Vorbereitungsregel bPR_{1_6} prüft ein auslösendes Link Ereignis auf eine Namensgleichheit zu einem eintretenden Link Ereignis. Namensgleiche Link Ereignisse stellen einen Ersatz für einen Sequenzfluss dar (Allweyer 2009, S. 80). Aus diesem Grund werden beide Ereignisse entfernt und der Vorgänger des auslösenden Link Ereignisses mit dem Nachfolger des eintretenden Link Ereignisses verknüpft. Im Gegensatz zur Prozessschnittstelle der eEPK werden auslösendes und empfangendes Link Ereignis im gleichen Prozessmodell verwendet. Eine Verknüpfung zu einem anderen Modell ist nicht vorgesehen.

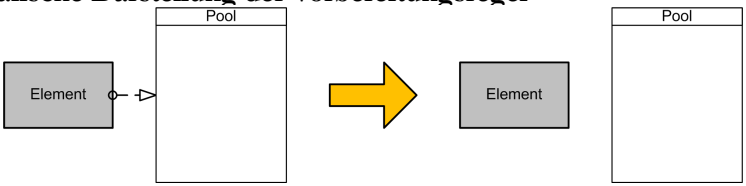
Grafische Darstellung der Vorbereitungsregel



Vorbereitungsregel bPR_{1_7}
If an element has an outgoing message flow that is connected to a pool, then the message flow is removed.

Die Vorbereitungsregel bPR_{1_7} prüft ein Element nach einem ausgehenden Nachrichtenfluss, der einen Pool als Zielpunkt hat. Dieses Konstrukt wird unter anderem bei öffentlichen Prozessen verwendet (Allweyer 2009, S. 57), die jedoch im Rahmen der Arbeit nicht betrachtet wird. Führt ein Nachrichtenfluss zu einem Pool, dann hat dieser keine Auswirkungen auf den Sequenzfluss (Freund et al. 2010, S. 101-102). Im Rahmen einer Simulation des modellierten Geschäftsprozesses kann der Nachrichtenfluss ignoriert werden und wird durch die Vorbereitungsregel entfernt.

Grafische Darstellung der Vorbereitungsregel

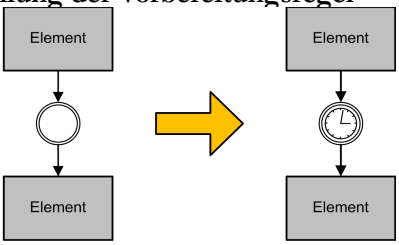


Vorbereitungsregel bPR_{1_7}
If the element is a catching intermediate event and it has not an incoming message flow, then it will be replaced by an intermediate timer event.

Die Vorbereitungsregel bPR_{1_11} identifiziert eintretende Zwischenereignisse, die keinen eingehenden Nachrichtenfluss aufweisen. Wenn ein solches Zwischenereignis in einer Sequenz durch eine Marke erreicht wird, dann wird der Prozessablauf erst fortgeführt, wenn das Ereignis eingetreten ist (Freund et al. 2010, S. 50). Bis das Ereignis eintritt, vergeht Zeit, in der der Prozess nicht weiter ausgeführt wird. Zur Abbildung dieser Zeitspanne im Rahmen eines Simulationsmodells wird das Ereignis mit einem Timer Zwischenereignis ersetzt. Für die

Anwendung der Vorbereitungsregel ist der Marker im eintretenden Zwischenereignis nicht von Bedeutung.

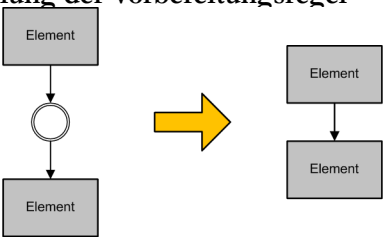
Grafische Darstellung der Vorbereitungsregel



Vorbereitungsregel bPR_{1_12}
If the element is a throwing intermediate event and it has not an outgoing message flow, then it will be removed.

Die Vorbereitungsregel bPR_{1_12} identifiziert ein auslösendes Zwischenereignis, welches keinen ausgehenden Nachrichtenfluss hat. Wenn eine Marke dieses Zwischenereignis erreicht, dann wird ein Ereignis vom entsprechenden Typ ausgelöst. Ein auslösendes Zwischenereignis ohne ausgehenden Nachrichtenfluss wird nicht weiter verarbeitet. Ohne diese Verarbeitung hat es keinen Einfluss auf die Ausführung des Prozesses. Das Zwischenereignis wird daher aus dem Modell entfernt.

Grafische Darstellung der Vorbereitungsregel



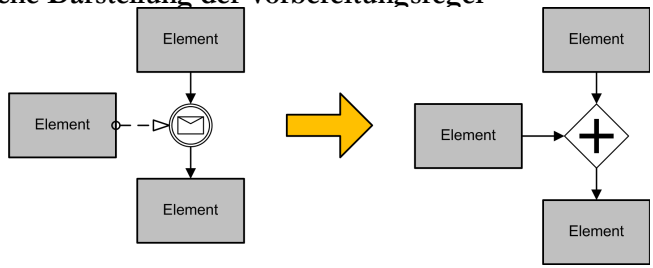
Vorbereitungsregelserie bPR_{N_x}

Die Vorbereitungsregelserie bPR_{N_x} wird durch die Vorbereitungsserie bPR_{2_x} aufgerufen.

Vorbereitungsregel bPR_{N_2}
If an intermediate message catching event has an incoming message flow, then it will be converted to a parallel gateway.

Die Vorbereitungsregel bPR_{N_2} adressiert ein Nachrichtenereignis, welches einen eingehenden Nachrichtenfluss aufweist. In diesem Fall wird der Sequenzfluss nur fortgeführt, wenn eine Nachricht eingetroffen ist. Semantisch wird von einem logischen Und ausgegangen. Dementsprechend wird das Ereignis in ein paralleles Gateway transformiert.

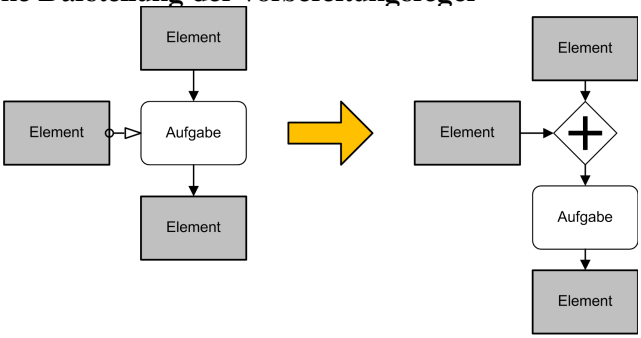
Grafische Darstellung der Vorbereitungsregel



Vorbereitungsregel bPR_{N_3}
If a task has an incoming message flow, then the task will get a parallel gateway as predecessor and the element that is connected with the message flow will get a predecessor of the gateway.

Die Vorbereitungsregel bPR_{N_3} prüft eine Aufgabe hinsichtlich eines eingehenden Nachrichtenflusses. Ausgeführt werden kann die Aufgabe nur, wenn eine Nachricht eingetroffen ist (Allweyer 209, S. 50). Dieser Zusammenhang kann als logisches Und aufgefasst werden. Die Aufgabe erhält ein paralleles Gateway als Vorgänger. Dieses hat als Vorgänger den Vorgänger der Aufgabe und das sendende Element des Nachrichtenflusses.

Grafische Darstellung der Vorbereitungsregel



Vorbereitungsregelserie bPR_{2_x}

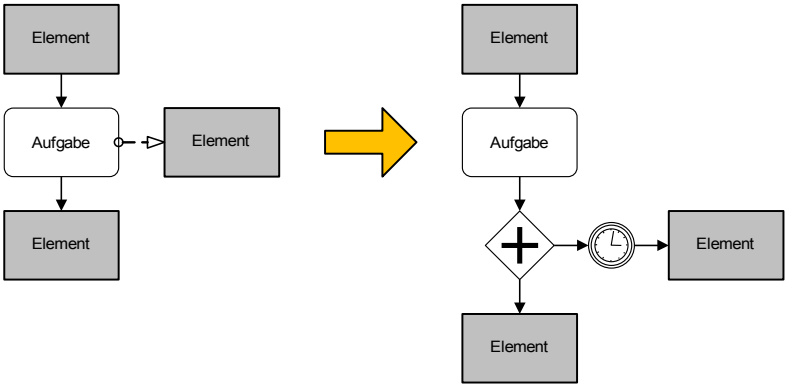
Vorbereitungsregel bPR_{2_3}
If a task has an outgoing message flow and not an incoming message flow from the same element, then the task will get a parallel gateway as successor and the gateway will get a timer event as a successor and this event will get the target of the message flow as successor and if the target of the message flow is a start event, then the rule bPR_{N_1} is executed or if the target is an intermediate event, then the rule bPR_{N_2} is executed or if the target is a task, then the rule bPR_{N_3} is executed.

Die Vorbereitungsregel bPR_{2_3} betrachtet Aufgaben, die einen ausgehenden Nachrichtenfluss haben und keinen eingehenden Nachrichtenfluss vom gleichen Element aufweist. Im Rahmen der Aufgabe wird in diesem Fall eine Nachricht versendet. Hierbei wird unterstellt, dass die Nachricht nicht während der Bearbeitung der Tätigkeit versendet wird, sondern nach deren Abschluss. Daher wird der Versand mittels eines parallelen Gateways, der als Nachfolger der Aufgabe eingeführt wird, realisiert. Die Übermittlungszeit der Nachricht wird durch einen Timer zwischen dem Gateway und dem Ziel des Nachrichtenflusses eingefügt.

Der Nachrichtenfluss wird im Zuge der Vorbereitungsregel in einen Sequenzfluss umgewandelt. Das Zielelement des Nachrichtenflusses würde dadurch keinen eingehenden Nachrichtenfluss mehr aufweisen. Aus diesem Grund wird durch die Vorbereitungsregel bPR_{2_3} eine Zusatzregel der Vorbereitungsserie bPR_{N_x} ausgeführt:

- Wenn das Ziel des Nachrichtenflusses ein Startereignis ist, dann wird Regel bPR_{N_1} angewendet.
- Wenn das Ziel des Nachrichtenflusses ein Zwischenereignis ist, dann wird Regel bPR_{N_2} angewendet.
- Wenn das Ziel des Nachrichtenflusses eine Aufgabe ist, dann wird Regel bPR_{N_3} angewendet.

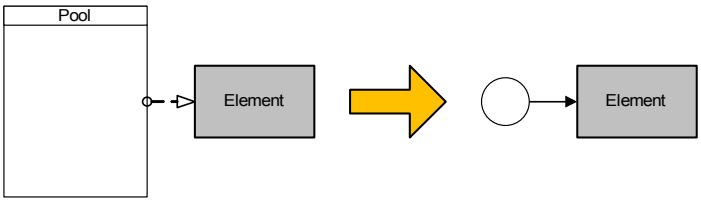
Grafische Darstellung der Vorbereitungsregel



Vorbereitungsregel bPR_{2_4}
If a pool has an outgoing message flow, then a source will become the source of the message flow and if the target of the message flow is a start event, then the rule bPR_{N_1} is executed or if the target is an intermediate event, then the rule bPR_{N_2} is executed or if the target is a task, then the rule bPR_{N_3} is executed.

Die Vorbereitungsregel bPR_{2_4} prüft einen Pool auf einen ausgehenden Nachrichtenfluss. Der Ursprung des Nachrichtenflusses ist nicht näher definiert. Semantisch wird angenommen, dass ein Nachrichtenfluss der von einem Pool ausgeht als Startpunkt und demnach als Startereignis aufgefasst werden kann. Während Nachrichtenflüsse die als Zielpunkt einen Pool haben mit Vorbereitungsregel bPR_{1_7} entfernt werden, haben vom Pool ausgehende Nachrichtenflüsse einen Einfluss auf die Ausführung des Prozesses. Entsprechend wird der Nachrichtenfluss in einen Sequenzfluss umgewandelt und die Quelle wird ein Startereignis. Die bPR_{N_x} Regeln des Zielelements sind äquivalent zur Vorbereitungsregel bPR_{2_3}.

Grafische Darstellung der Vorbereitungsregel

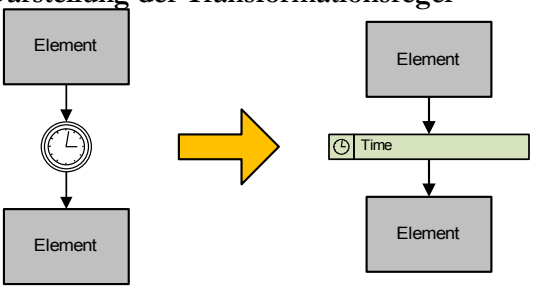


Transformationsregeln – bTR (BPMN Transformation Rules)

Transformationsregel eTR_4
If the element is an intermediate timer event, then it will be a delay.

Die Transformationsregel bTR_4 überführt ein Timer Ereignis in ein Delay. Während ein Timer Ereignis auch einen Zeitpunkt darstellen kann (Freund et al. 2010, S. 56), repräsentiert das Delay immer eine Zeitspanne. Dies ist den Anforderungen an ein Simulationsmodell geschuldet, bei dem mit Zeitspannen gearbeitet wird. Das Timer Event stellt auch das einzig verbleibende Zwischenereignis dar, welches nicht durch die Vorbereitungsregeln entfernt oder in dieses überführt wurde.

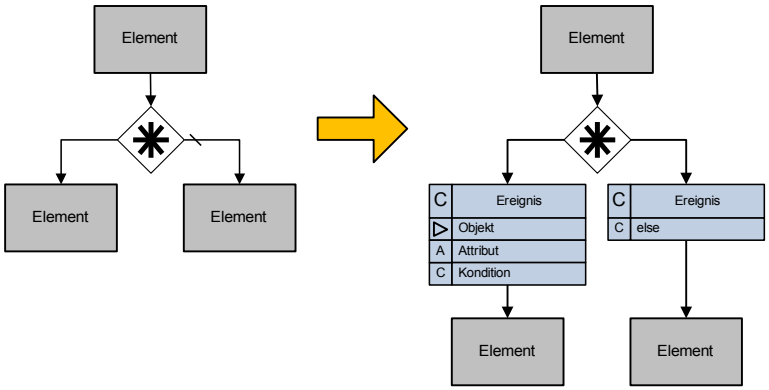
Grafische Darstellung der Transformationsregel



Transformationsregel eTR_{11}
If a complex gateway has more than one successor, then it will be a complex split gateway and every successor of the gateway will get an exclusive key as predecessor that is the successor of the split gateway and if the complex gateway has a standard flow, then the corresponding complex key will be marked with an else.

Die Transformationsregel bTR_{11} überführt ein komplexes Gateway mit mehreren Nachfolgern in das ProSiT Ablaufdiagramm. Durch die Transformation werden komplexe Schlüssel als Nachfolger der Gateways eingefügt. Diese enthalten die Bedingungen, die erfüllt sein müssen, damit der Prozesspfad ausgeführt wird (Allweyer 2009, S. 36). Aus semantischer Sicht kann das Gateway auch als inklusives Oder aufgefasst werden (Freund et al. 2010, S. 41). Welcher Prozesspfad auszuführen ist, wird durch Attribute eines Markers bestimmt, die das Prozessmodell durchlaufen. Entsprechend muss das zu prüfende Attribute mit Werten hinterlegt werden. Wenn ein Nachfolger aber mit einem Standardfluss verbunden ist, dann wird der entsprechende Pfad des komplexen Gateways mit einem else-Schlüssel versehen. Mindestens ein Prozesspfad wird durch diesen else-Schlüssel ausgeführt.

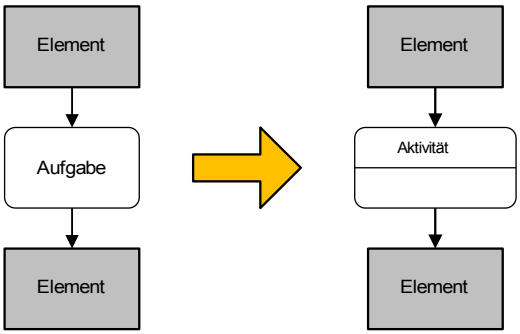
Grafische Darstellung der Transformationsregel



Transformationsregel eTR_{13}
If the element is a task, then it will be an activity and if the task is placed in a lane, then the deepest lane is added as a resource and if the task has boundary events, these events will be added to the activity.

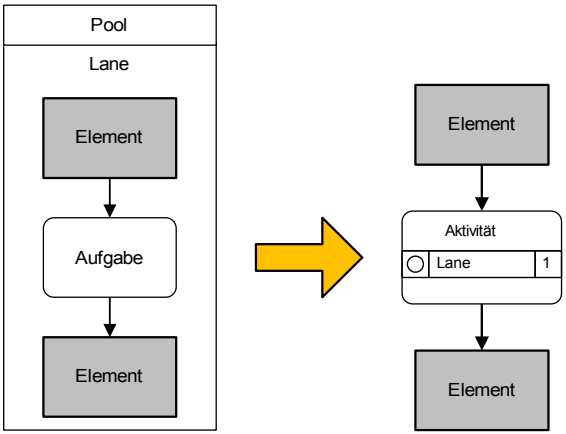
Die Transformationsregel bTR_{13} überführt eine Aufgabe in eine Aktivität. Die Bezeichnung der Aufgabe wird als Tätigkeit der Aktivität übernommen. Neben dieser Hauptregel definiert die Transformationsregel noch zwei Zusatzregeln, die Ressourcen und angeheftete Ereignisse betrachtet.

Grafische Darstellung der Transformationsregel



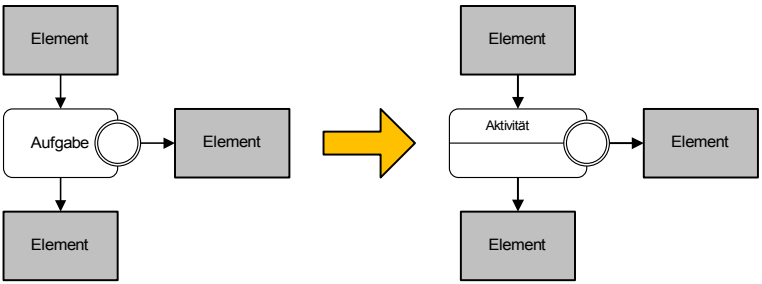
Zusatzbedingung für die Einbindung in einer Lane

Die erste Zusatzbedingung prüft, ob eine Aufgabe in einer Lane eingebunden ist. In diesem Fall ist, dann wird die am tiefsten verschachtelte Lane (Freund et al. 2010, S. 46) als Ressource der Aktivität hinzugefügt.



Zusatzbedingung für angeheftete Ereignisse

Angeheftete Ereignisse können während der Ausführung einer Aufgabe eintreten, womit entweder eine weitere Marke dem Prozessmodell hinzugefügt oder die Ausführung der Aufgabe unterbrochen wird. Diese Ereignisse werden ebenfalls an das ProSiT Ablaufdiagramm übergeben, damit diese im Rahmen der Normalisierung weiter verarbeitet werden können.



3.4.3 Vom UML Aktivitätsdiagramm zum Transformationsmodell

In diesem Kapitel werden die Transformationsregeln erläutert, um ein UML Aktivitätsdiagramm in das ProSiT Ablaufdiagramm zu überführen. Im Gegensatz zur eEPK, bei der die konkreten Vorgänger und Nachfolger bei den Transformationsregeln beachtet werden müssen, ist es beim UML Aktivitätsdiagramm ausreichend, die Anzahl der Vorgänger und Nachfolger zu betrachten. Aufgrund der möglichen Anzahl an Kombinationen, die in Tabelle 7 aufgeführt werden, ist eine entsprechend große Anzahl an Vorbereitungsregeln notwendig.

Syntaxregeln – adSR (Activity Diagram Syntax Rule)

Grundsätzlich sind die in den Spezifikationen des UML Aktivitätsdiagramms aufgeführten Einschränkungen zu beachten (OMG 2010). Hinsichtlich der erlaubten Kombinationen der Elemente wird auf Tabelle 7 sowie dessen Erläuterung, verwiesen. Wenn ein Modell diese Bedingungen erfüllt, dann kann es mit den aufgeführten Transformationsregeln überführt werden.

Eine Einschränkung liegt jedoch vor, wenn ein Bereich verwendet wird, der durch eine Ausnahme abgebrochen werden kann. Wenn in einem solchen Bereich eine Gabelung oder Vereinigung verwendet wird, dann kann das Modell nicht überführt werden. Der Grund hierfür liegt in der fehlenden Synchronisation zwischen parallel laufenden Prozesspfaden. So gibt es in den betrachteten Simulationsumgebungen keine Möglichkeit parallel laufende Aktivitäten durch eine Ausnahme zeitgleich zu unterbrechen. Aus diesem Grund wird eine entsprechende Einschränkung vorgegeben, die in Form der Syntaxregel $adSR_1$ definiert wird.

Syntaxregel $adSR_1$
If an expansion region has an accept event action with an outgoing exception handler and the expansion region contains a fork node or a join node, than the model can not be converted to the ProSiT sequence diagram.

Vorbereitungsregeln – adPR (Activity Diagram Preparation Rule)

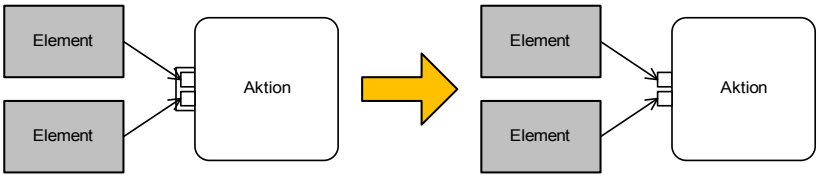
Die Vorbereitungsregeln für das UML Aktivitätsdiagramm basieren auf einem zweistufigen Konzept. Zuerst werden die Vorbereitungsregeln $adPR_{1\_x}$ ausgeführt. Anschließend erfolgt die Anwendung der $adPR_{2\_x}$ Vorbereitungsregeln, die teilweise auf den Ergebnissen der ersten Serie basieren. Diese Unterteilung ermöglicht die Reduktion der Anzahl der notwendigen Vorbereitungsregeln. Einige Vorbereitungsregeln der $adPR_{2\_x}$ Serie beziehen sich auch auf mehrere Elemente. Hierdurch werden beispielsweise fünf Vorbereitungsregeln, die sich auf jeweils ein Element beziehen würden, in einer Regel zusammengefasst.

Vorbereitungsregelserie $adPR_{1\_x}$

Vorbereitungsregel $adPR_{1\_2}$
If an action has a parameter set with incoming process flows, then the parameter set is removed.

Die Vorbereitungsregel $adPR_{1\_2}$ identifiziert eine Aktion, die nur über eine Parametergruppe mit eingehenden Prozesspfaden verfügt. Aus semantischer Sicht entspricht die Aktion einer Aktion mit mehreren Objektknotenpins, welche die Vorgänger der Aktion beinhalten. Aus diesem Grund wird die Parametergruppe entfernt.

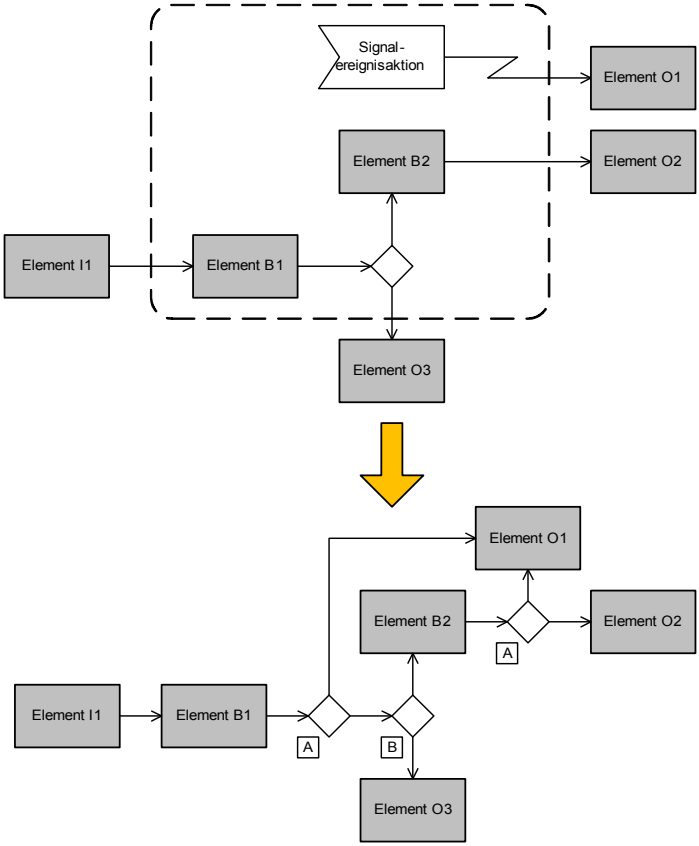
Grafische Darstellung der Vorbereitungsregel



Vorbereitungsregel adPR$_{1_4}$
If an expansion region contains a signal event action that has not a predecessor and an outgoing exception handler, then the expansion region and the signal event action are removed and all elements that are not a decision node will get a decision node as successor that has a first successor the successor of the element and as a second successor the element that is linked with the exception handler.

Die Vorbereitungsregel adPR$_{1_4}$ prüft bei einem Bereich, ob dieser eine Signalereignisaktion beinhaltet, die keinen Vorgänger und eine ausgehende Ausnahmebehandlung aufweist. Wird dieses Signal ausgelöst, dann werden alle laufenden Aktionen im Bereich abgebrochen werden (Oestereich 2006, S. 314-315). Für diese Realisierung in einem Simulationsmodell wird dieses Konstrukt in ein vereinfachtes Modell umgewandelt. Zunächst werden der Bereich und die Signalereignisaktion entfernt. Auf jedes Element, welches keine Verzweigung (B) ist, erhält eine Verzweigung (A) als Nachfolger, welche die Ausnahmebehandlung abdeckt. Verzweigungen sind dabei ausgeschlossen. Diese nehmen nach dem Abschluss einer Aktivität keine Zeit in Ansprung, um den nachfolgenden Pfad zu wählen. Würden diese berücksichtigt gäbe es eine doppelte Prüfung, welche die Ausführung abbricht. Der erste Nachfolger der Verzweigung (A) ist der Nachfolger des Elements, welches die Verzweigung (A) als Nachfolger erhalten hat. Als zweiten Nachfolger erhält die Verzweigung (A) das Element, dass mit der Signalereignisaktion durch die Ausnahmebehandlung verknüpft wurde. Durch diese Realisierung kann ein Abbruch innerhalb des entfernten Bereichs modelliert werden, jedoch mit einer Einschränkung. Ein Abbruch kann nur nach dem Abschluss eines Elements realisiert werden.

Grafische Darstellung der Vorbereitungsregel



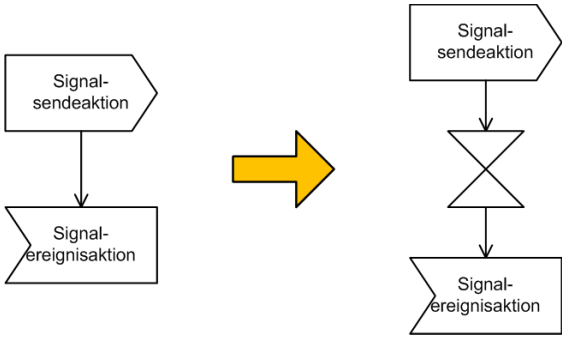
Nach Oestereich (2006, S. 315)

Vorbereitungsregel adPR$_{1_6}$
If a signal send action has a signal event action as a predecessor and the model does not contain a signal event action with the same name as the signal send action, than the connection between those two elements is removed and a new time event is added to the model and the send signal action will get the time event as its successor and the accept event action will get the time event as its predecessor.

Die Vorbereitungsregel $adPR_{1\_6}$ prüft, ob eine Signalsendeaktion eine Signalereignisaktion als Nachfolger hat. Zusätzlich gilt die Prämisse, dass es im Modell keine Signalereignisaktion gibt, die den gleichen Namen wie die Signalsendeaktion aufweist. Dieser Fall wird in Vorbereitungsregel $adPR_{1\_5}$ verarbeitet. Bei diesem Konstrukt wird aus semantischer Sicht geschlussfolgert, dass eine Nachricht über die Signalsendeaktion versendet wird und die Signalereignisaktion auf das Ergebnis, die Rückantwort, des gesendeten Signals wartet. Ein Beispiel wäre eine gesendete Anfrage. Der Prozess wird erst fortgesetzt, wenn eine Rückantwort in Form eines Angebots vorliegt (siehe auch Overhage et al. 2011, S. 750). Dass immer eine Rückantwort gesendet wird, ist bei diesen Vorbereitungsregeln eine unterstellte Prämisse.

Das Senden und Beantworten des gesendeten Signals benötigt eine gewisse Zeit. Diese wird mit einem Zeitereignis zwischen der Signalsendeaktion und Signalereignisaktion berücksichtigt.

Grafische Darstellung der Vorbereitungsregel

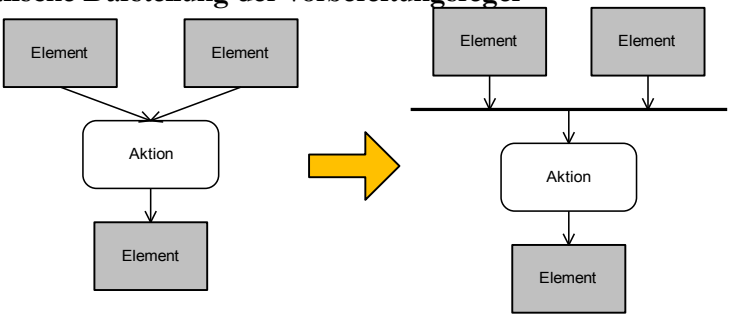


Vorbereitungsregelserie $adPR_{2\_x}$

Vorbereitungsregel $adPR_{2\_8}$
If the element is an action or an accept event action or a send signal action or a call action or a time event or an expansion region and the element more than one predecessor and a successor, then the element will get a join node as a predecessor and every predecessor of the element will become the predecessor of the join node.

Die Vorbereitungsregel $adPR_{2\_8}$ identifiziert ein Element vom Typ Aktion, Signalereignisaktion, Signalsendeaktion, Aufrufaktion, Zeitereignis oder Bereich, dass einen Vorgänger und mehr als einen Nachfolger hat. Durch das Vorhandensein von Vorgängern und eines Nachfolgers ist das Element in den Prozessfluss eingebunden. Damit dieses ausgeführt werden kann, müssen alle Vorgänger abgeschlossen sein (Arlow und Neustadt 2005, S. 293). Vor das Element wird hierfür eine Vereinigung eingefügt und diese mit jedem Vorgänger verknüpft.

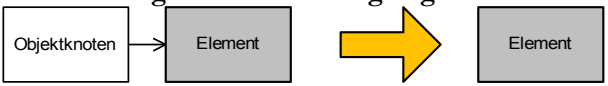
Grafische Darstellung der Vorbereitungsregel



Vorbereitungsregel $adPR_{2\_16}$
If an object note or a data store has not a predecessor and it has a successor, that it is removed.

Die Vorbereitungsregel $adPR_{2\_16}$ identifiziert einen Objektknoten oder einen Datenspeicher, der keinen Vorgänger und einen Nachfolger hat. Das Element stellt in diesem Falle ein Input Parameter dar (Arlow und Neustadt 2005, S. 293). Für die Simulation wird unterstellt, dass alle notwendigen Daten zur Verfügung stehen. Daher wird das Element aus dem Modell entfernt.

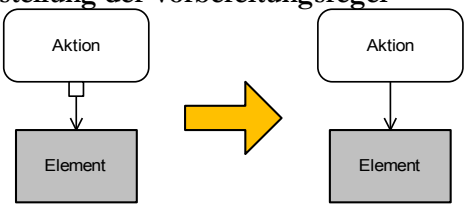
Grafische Darstellung der Vorbereitungsregel



Vorbereitungsregel $adPR_{2\_22}$
If an object node pin has a successor, then it will be removed.

Die Vorbereitungsregel $adPR_{2\_22}$ identifiziert einen Objektknoten, der als Pin einer Aktion angehängt ist und der einen Nachfolger hat. Dieses Konstrukt ist eine andere Darstellung eines Objektflusses (Oestereich 2006, S. 310). Da Objekte im Rahmen der Simulation nicht berücksichtigt werden, wird dieser Objektknoten entfernt und die Aktion direkt mit dem Nachfolger verknüpft.

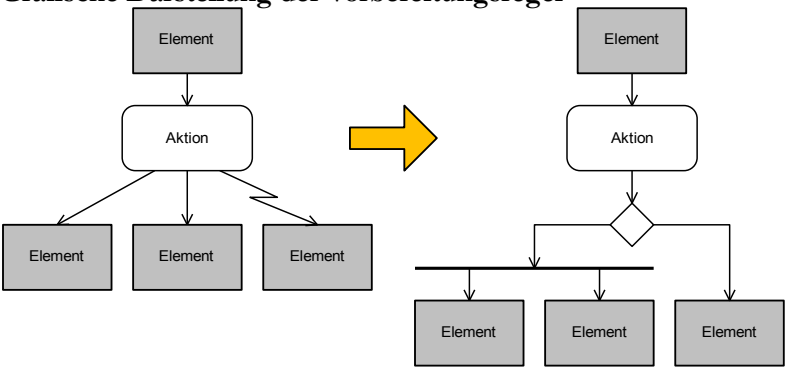
Grafische Darstellung der Vorbereitungsregel



Vorbereitungsregel $adPR_{2\_27}$
If an action has more than one successor and an exception handler, then the action gets a decision node as successor and a new fore node, that will get the successors of the action as its successors and exception handler will become successors of the decision node.

Die Vorbereitungsregel $adPR_{2\_27}$ identifiziert, eine Aktion, die eine Ausnahmebehandlung und mehr als einen Nachfolger vorweist. Wenn bei der Ausführung der Aktion eine Ausnahme eintritt, dann wird lediglich der Prozesspfad der Ausnahmebehandlung ausgeführt (Oestereich 2006, S. 311). Hierfür erhält die Aktion eine Verzweigung als Nachfolger. Tritt die Ausnahme nicht ein, dann werden alle anderen Nachfolger ausgeführt (Oestereich 2006, S. 310). Diese werden mittels einer Gabelung an die Verzweigung angefügt.

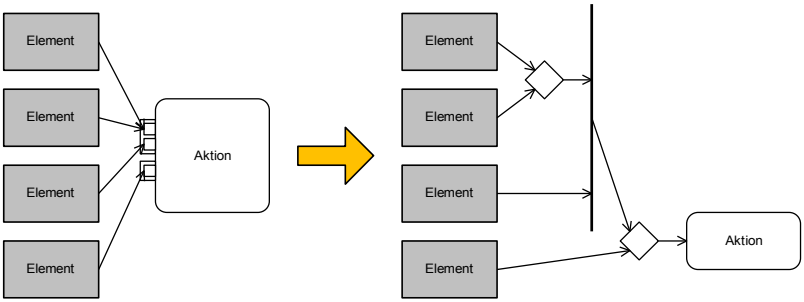
Grafische Darstellung der Vorbereitungsregel



Vorbereitungsregel $adPR_{2\_28}$
If an action has more than one parameter set with incoming process flows, then the action will get a merge node as its predecessor and if a parameter set has one object node pin, then it will become the predecessor of the merge node and if a parameter set has more than one object node pin, then they will get a join node as successor and the join node will become a predecessor of the merge node and if more than one element has an object node pin as their successors than these elements will get an additional merge node has their common successor.

Die Vorbereitungsregel $adPR_{2\_28}$ betrachtet Parametergruppen mit eingehenden Prozesspfaden. Mehrere Parametergruppen bei einer Aktion stehen in einer exklusiven Oder Beziehung. Mehrere Objektknotenpins in einer Parametergruppe hingegen sind mit einem logischen Und verknüpft (Arlow und Neustadt 2005, S. 321). Aus diesem Grund wird, wenn mehr als eine Parametergruppe vorliegt, der Aktion eine Zusammenführung als Vorgänger vorangestellt. Wenn in einer Parametergruppe nur ein Objektknotenpin vorhanden ist, dann wird diese der direkte Nachfolger der Zusammenführung. Wenn jedoch eine Parametergruppe mehr als einen Vorgänger hat, dann wird eine Vereinigung eingefügt, die Vorgänger der Zusammenführung ist und als eigene Vorgänger die Elemente bekommt, die auf die Objektknotenpins der Parametergruppe verweisen. Zusätzlich wird geprüft, ob mehr als ein Element Vorgänger eines Objektknotenpins ist. In diesem Falle wird zwischen die Elemente und der Vereinigung eine zusätzliche Zusammenführung eingefügt. Die Summe aller Möglichkeiten, die in dieser Regel berücksichtigt werden, sind in der grafischen Darstellung aufgeführt.

Grafische Darstellung der Vorbereitungsregel



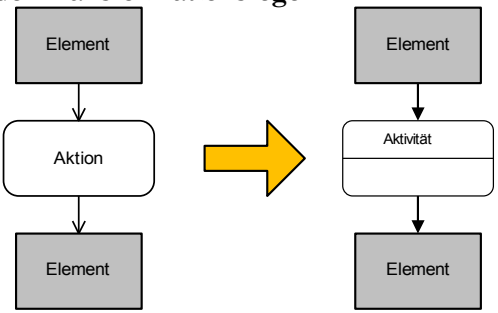
Transformationsregeln – adTR (Activity Diagram Transformation Rules)

Die nachfolgend beschriebenen Transformationsregeln überführen das durch die Vorbereitungsregeln präparierte UML Aktivitätsdiagramm in das ProSiT Ablaufdiagramm.

Transformationsregel adTR$_7$
If the element is an action, then it will be an activity and if the action is placed in a partition, then the deepest partition is added as a resource or if the action contains the information about a partition, then the deepest partition is added as a resource.

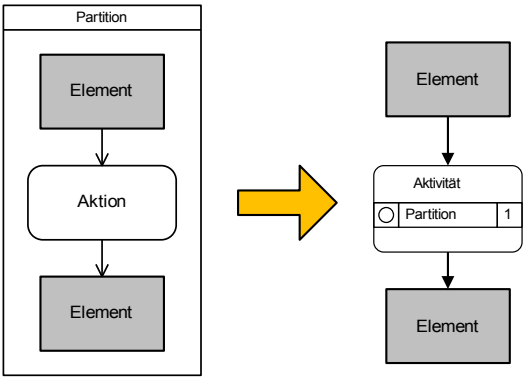
Die Transformationsregel adTR$_7$ überführt eine Aktion in eine konzeptuelle Aktivität. Die Bezeichnung der Aktion wird der Aktivität als Tätigkeit übergeben. Die im UML Aktivitätsdiagramm verwendeten Partitionen werden als Ressourcen aufgefasst und der Aktivität hinzugefügt. Diese können in zwei Arten vorliegen: Entweder als grafische Elemente dem UML Aktivitätsdiagramm oder direkt in der Aktion eingetragen (OMG 2010, S. 342-343).

Darstellung der Transformationsregel



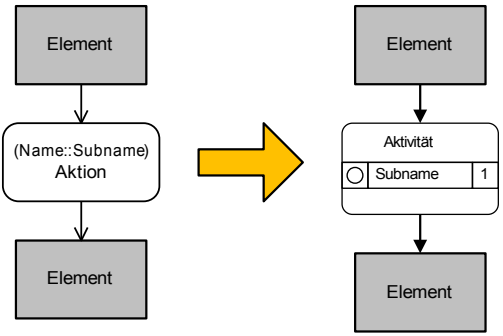
Partition als grafisches Element

Die erste Zusatzregel betrachtet die Partition als grafisches Element. Hierbei wird die am tiefsten aufgeführte Partition (OMG 2011, S. 344) als Ressource der Aktivität hinzugefügt.



Partition als Teil der Bezeichnung der Aktion

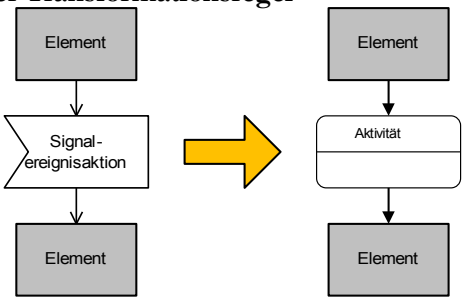
Die zweite Zusatzregel betrachtet die Partition als Teil der Bezeichnung der Aktion. Während bei der obigen Abbildung nur eine Partition aufgeführt wird, liegt bei der nachfolgenden Abbildung eine geschachtelte Partition vor. Hierbei gilt, dass nur die tiefste Partition als Ressource aufgenommen wird.



Transformationsregel $adTR_8$
If the element is a signal event action, then it will be an activity and if the signal event action is placed in a partition, then the deepest partition is added as a resource.

Die Transformationsregel $adTR^8$ überführt eine Signalereignisaktion in eine Aktivität. Eine Signalereignisaktion wird ebenfalls als Tätigkeit aufgefasst, die an einem Objekt von Ressourcen ausgeführt wird. Diese Interpretation ist auch bei Overhage (2011, S. 749-750) vorzufinden, bei der die Signalereignisaktion wie eine Funktion in der eEPK aufgefasst wird. Im Gegensatz zur Aktion, werden in einer Signalereignisaktion keine Partitionen aufgeführt. Daher ist nur die erste Zusatzregel aus der Transformationsregel $adTR_7$ in der Transformationsregel $adTR_8$ enthalten. Entsprechend wir für die grafische Darstellung der Signalereignisaktion in einer Partition auf Transformationsregel $adTR_7$ verwiesen.

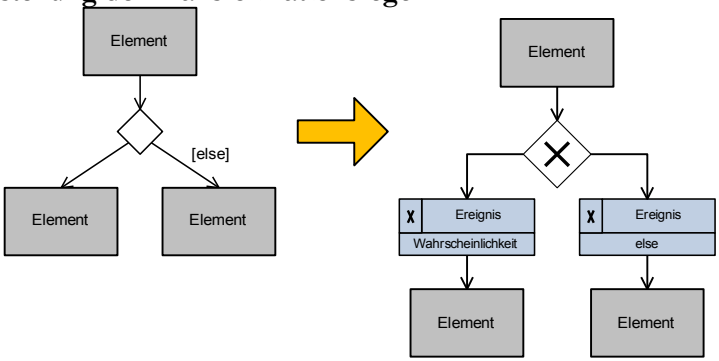
Darstellung der Transformationsregel



Transformationsregel $adTR_{12}$
If the element is a decision node, then it will be an exclusive Gateway and every successor will get an exclusive key as successor that will be the successor of the exclusive Gateway and if a control flow is marked with an else, then the corresponding exclusive key will be marked with an else.

Die Transformationsregel $adTR_{12}$ überführt eine Verzweigung in ein verzweigendes exklusives Gateway. Gemäß der Spezifikationen der OMG (2009, S. 360) kann jede eintreffende Marke nur über einen ausgehenden Pfad weitergeleitet werden. Daraus resultiert, dass beide Elemente eine exklusive Entscheidung repräsentieren, wodurch das exklusive Gateway einen exklusiven Schlüssel als Nachfolger erhält. Die Zusatzbedingung prüft darüber hinaus einen Kontrollfluss im UML Aktivitätsdiagramm, ob dieser als else-Zweig markiert wurde. Der korrespondierende exklusive Schlüssel wird entsprechen mit einem else markiert. Sind an den Kanten Bezeichnungen, so werden diese als Ereignis in den exklusiven Schlüsseln verwendet.

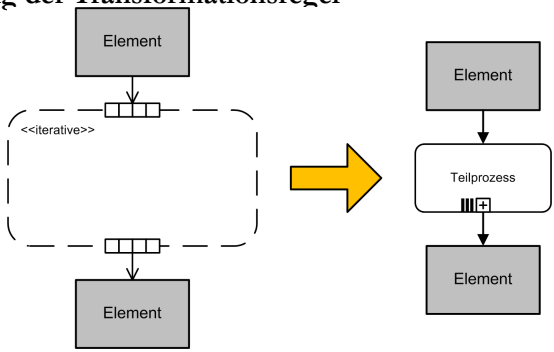
Darstellung der Transformationsregel



Transformationsregel adTR$_{16}$
If an expansion region is marked as iterative, then it will be a sub process marked with a sequential marker.

Die Transformationsregel adTR$_{16}$ prüft eine Region nach dem Schlüsselwort „iterative“. Alle Aktionen in diesem Bereich werden mehrfach sequenziell ausgeführt. Zur Repräsentation dieses Konstrukts wird der Bereich in einen Teilprozess überführt, der die einzelnen Elemente des Bereichs beinhaltet. Der Inhalt des Bereichs wird als eigenständiger Teilprozess aufgefasst, auf den Vorbereitungs- und Transformationsregeln anzuwenden sind. Für die sequenzielle Ausführung wird der Teilprozess mit einem sequenziellen Marker versehen.

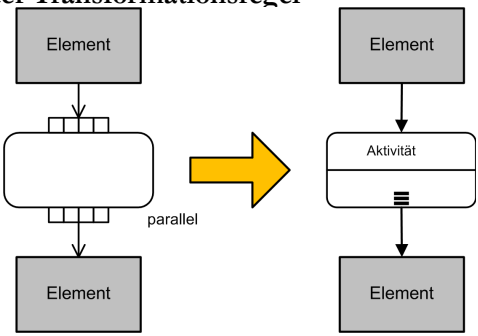
Darstellung der Transformationsregel



Transformationsregel $adTR_{19}$
If an expansion region is modeled in the shorthand notation, than it will be an activity with a parallel marker and if the expansion region is placed in a partition, then the deepest partition is added as a resource.

Die Transformationsregel $adTR_{19}$ zielt auf verkürzte Bereiche ab. Diese können auf unterschiedlichste Weise modelliert werden (OMG 2010, S. 384; Oestereich 2011, S. 316). Die verkürzten Bereiche beinhalten nur eine Aktivität, die mehrfach ausgeführt wird. Hierbei ist zu beachten, dass die Ausführung parallel erfolgt (OMG 2010, S. 384). Die auszuführende Aktion wird direkt als Bezeichnung des Bereichs verwendet. Entsprechend wird das Element in eine Aktivität mit einem parallel Marker überführt. Wenn der Bereich sich darüber hinaus in einer Partition befindet, dann wird wie bei Transformationsregel $adTR_7$, die tiefste Partition als Ressource in der Aktivität aufgenommen.

Darstellung der Transformationsregel



3.5 Die Normalisierung des Transformationsmodells

Die Normalisierung des Transformationsmodells besteht aus drei Teilbereichen: automatische und semiautomatische Normalisierungsregeln sowie der manuellen Normalisierung. Ähnlich der Transformationsregeln werden Normalisierungsregeln in Form einer Wenn-Dann Hypothese formuliert. Ist diese in englischer Sprache, dann ist die Regel unabhängig von der verwendeten Sprache in den Notationselementen. In deutsch konzipierte Regeln greifen auf sprachliche Elemente zurück. Eine Überführung in eine andere Sprache muss entsprechend geprüft werden. Anschließend wird die Normalisierungsregel erläutert und abstrakt dargestellt.

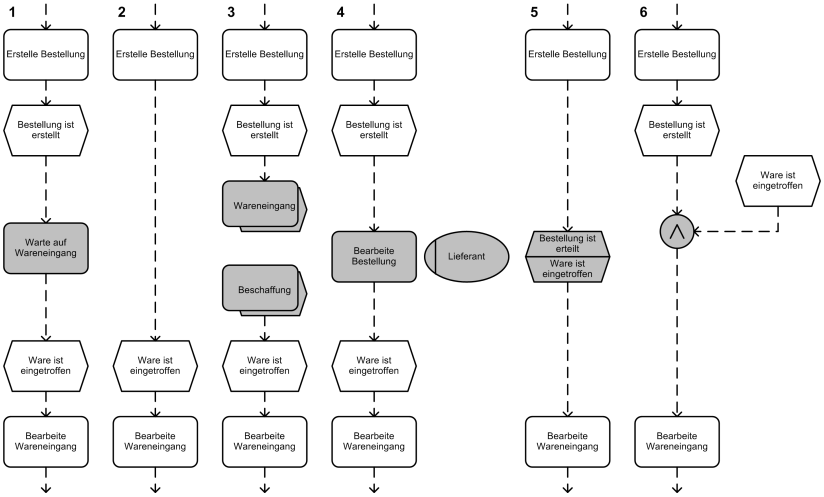
3.5.1 Automatische Normalisierungsregeln

Normalisierungsregel aNR_1
If a parallel joining pathway has two predecessors and one of them is a source, then the source will be removed and the pathway will be replaced by a delay.

Die Normalisierungsregel aNR_1 geht auf Überlegungen von Rosemann (1996, S. 210-212) zurück. Er erarbeitete Möglichkeiten um externe Funktionen in einer EPK zu modellieren, die außerhalb des Betrachtungsbereichs stattfinden (Abbildung 44). Rosemann (1996, S. 94-104) bewertet die Alternativen nach den Grundsätzen ordnungsmäßiger Modellierung. Die zu priorisierende Darstellungsform sei die sechste Alternative.

Bezogen auf den dargestellten Ablauf wird das Ereignis „Ware ist eingetroffen“, durch die Transformationsregel eTR_1, zu einer Quelle. Ablauftechnisch könnte die Ware eintreffen, noch bevor die Bestellung ausgelöst wurde. Hintergedanke bei externen Funktionen ist eine Zeitdauer, die keine eigenen Ressourcen beansprucht. Um diesen Zeitaspekt zu berücksichtigen und um den möglichen Logikfehler zu vermeiden, wandelt Normalisierungsregeln aNR_1 dieses Muster in die sechste Alternative. Während eine Wartefunktion in einer EPK

abzulehnen ist (Rosemann 1996, S. 210), kann im ProSiT Ablaufdiagramm eine prozessbedingte Wartezeit mit einer Verzögerung realisiert werden.

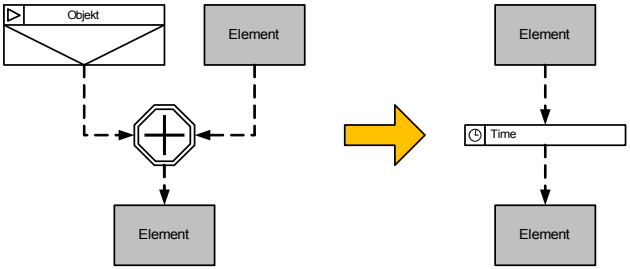


Rosemann (1996, S. 211)

Abbildung 44: Alternative Möglichkeiten für externe Funktionen in der EPK

Das zusammenführende parallele Gateway sowie die Quelle werden in eine Verzögerung, welche eine Wartezeit realisiert, die keine Ressourcen beansprucht.

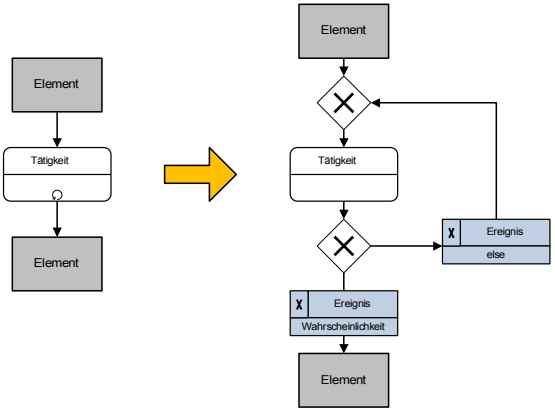
Darstellung der Normalisierungsregel



Normalisierungsregel aNR_2
If an activity has a loop marker, then the marker is removed and the activity will get a joining exclusive gateway as its predecessor and a splitting exclusive gateway as its successor. The splitting gateway will get an exclusive key as a first successor that will get the successor of the activity as its successor and as a second successor an exclusive else key that has the joining exclusive gateway as its successor.

Die Normalisierungsregel aNR_2 überführt eine konzeptuelle Aktivität mit einem Schleifen Marker in eine Aktivität, die über weitere Normalisierungsregeln zu einer konsistenten Aktivität transformiert werden kann. Durch den Schleifen-Marker wird die Aktivität für eine unbestimmte Anzahl an Durchläufen ausgeführt, bis ein Abbruchkriterium erfüllt ist (OMG 2011, S. 190). Um diesen semantischen Sachverhalt ohne den Schleifen-Marker darzustellen, wird die Aktivität in eine syntaktische Schleife mit exklusiven Gateways überführt. Hierfür erhält die Aktivität ein zusammenführendes exklusives Gateway als Vorgänger und ein Verzweigendes als Nachfolger. Der erste Schlüssel des verzweigenden Gateways erhält die Abbruchbedingung in Form einer Wahrscheinlichkeit. Wird dieser Schlüssel nicht ausgelöst, wird die Marke über einen exklusiven else-Schlüssel an das zusammenführende Gateway überreicht. Durch dieses Konstrukt wird das Verhalten des Schleifen-Markers abgebildet, womit dieser entfernt werden kann.

Darstellung der Normalisierungsregel



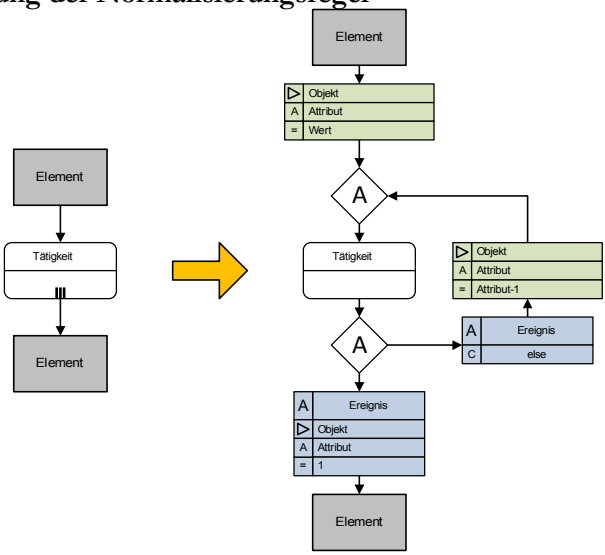
Normalisierungsregel aNR_3
If an activity has a sequence multi-instance marker, then the marker is removed and the activity will get a joining attribute-based gateway as its predecessor and a splitting attribute-based gateway as its successor. The splitting gateway will get as a first successor an attribute-based key that will get the successor of the activity as its successor and as a second successor an attribute-based else key that will get an additional attribute set as its successor and the attribute set will get the joining attribute-based gateway as its successor. The joining gateway will get an attribute set as its predecessor and this element will get the predecessor of the activity as its predecessor.

Die Normalisierungsregel aNR_3 überführt analog zur Normalisierungsregel aNR_2 eine Aktivität mit einem „Sequence Multi-Instance" Marker in eine normalisierbare Aktivität. Anstelle von exklusiven Gateways werden jedoch attributsbasierte Gateways verwendet. Zusätzlich wird aber zwischen dem Vorgänger der Aktivität und dem zusammenführenden Gateway eine Attributsfestlegung eingefügt. Das nachfolgende verzweigende Gateway erhält zwei Schlüssel. Tritt der Schlüssel mit Attributswert ein, wird der ursprüngliche Prozessablauf fortgesetzt. Der else-Schlüssel hingegen hat als Nachfolger eine weitere Attributsfestlegung, um die mehrfache Ausführung zu steuern.

Hierfür bedarf das Objekt ein Attribut. Durch die Anwendung der Normalisierungsregel wird das Attribut dem Objekt in der Objektsicht hinzugefügt. Als Bezeichnung wird „sequenceMultiInstance“, gefolgt von einer durchlaufenden Nummer in Form einer natürlichen Zahl verwendet.

Dieses Attribut wird für die Attributsfestlegungen und attributsbasierten Schlüssel verwendet. Die erste Attributsfestlegung bestimmt wie folgt die nachfolgende Aktivität auszuführen ist. Nach der Ausführung der Aktivität erfolgt die Prüfung, ob der Wert 1 erreicht wurde. In diesem Falle würde der eigentliche Prozessablauf fortgesetzt. Andernfalls wird der else-Schlüssel ausgelöst, womit das Attribut um den Wert 1 reduziert wird und die Aktivität erneut ausgeführt wird. Diese Umsetzung realisiert eine nachgelagerte Prüfung, wie diese auch in den BPMN Spezifikationen (OMG 2011, S. 190) für eine Schleife definiert ist.

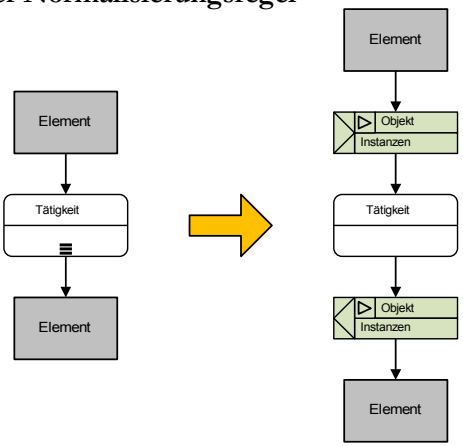
Darstellung der Normalisierungsregel



Normalisierungsregel aNR_4
If an activity has a sequence multi-instance marker, then the marker is removed and the activity will get a branch instance as predecessor and a merge instance as a successor.

Normalisierungsregel aNR_4 wandelt den „Parallel Multi-Instance“ Marker einer Aktivität in andere Elemente um, die den gleichen semantischen Sachverhalt auszudrücken. Eine Aktivität mit diesem Marker kann unabhängig voneinander parallel ausgeführt werden. Zur Realisierung dieses Verhaltens müssen mehrere Marken die Aktivität auslösen. Dies wird durch eine erzeugende Instanziierung als Vorgänger und eine zusammenführende Instanziierung als Nachfolger der Aktivität ermöglicht. Da im Rahmen von Simulationsmodellen beide Elemente unabhängig voneinander sind, muss die Anzahl der parallelen Ausführungen, im Gegensatz zur sequenziellen Mehrfachverarbeitung, fest vorgegeben werden. Andersfalls können nicht alle erzeugten Kopien der Marke wieder mittels einer Synchronisation vereinigt werden.

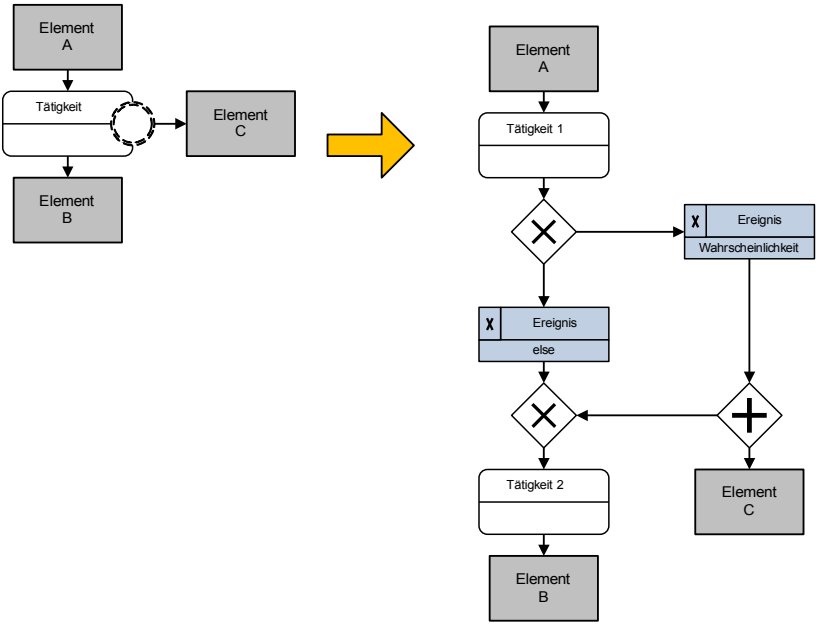
Darstellung der Normalisierungsregel



Normalisierungsregel aNR_5
If an activity has a non-interrupting boundary event, then the activity is divided in two activities. The first activity will get a splitting exclusive gateway as successor. This exclusive gateway will get two successors, an exclusive key and an exclusive else key. The exclusive else key will get a joining exclusive gateway as successor and the exclusive key will get a splitting parallel gateway as successor. The parallel gateway has as a first successor the successor of the boundary event and as a second successor the joining exclusive gateway. The joining exclusive gateway will get the second activity as successor and the activity will get the original successor of the activity as successor.

Normalisierungsregel aNR_5 überführt ein angeheftetes nicht unterbrechendes Ereignis in einfachere Elemente. Wenn das angeheftete Ereignis eintritt, dann wird parallel zur Aktivität ein weiterer Prozessablauf gestartet. Damit dieses Verhalten in unterschiedlichen Simulationsumgebungen umgesetzt werden kann, wird die Aktivität zweigeteilt; in Aktivität 1 und Aktivität 2. Die erste Aktivität erhält ein verzweigendes exklusives Gateway als Nachfolger. Mit diesem erfolgt die Prüfung, ob das angeheftete Ereignis eintritt. Die prozentuale Wahrscheinlichkeit des Eintretens wird im exklusiven Schlüssel hinterlegt, das Nichteintreten im exklusiven else-Schlüssel. Wenn das Ereignis eintritt, wird die Aktivität nicht unterbrochen, womit weiterhin die Aktivität 2 ausgeführt werden muss. Hierfür folgt auf den exklusiven Schlüssel ein verzweigendes paralleles Gateway. Dieses hat als Nachfolger den Nachfolger des angehefteten Ereignisses, womit der zusätzliche Prozesspfad ausgeführt werden kann. Als zweiten Nachfolger hat das Gateway ein zusammenführendes exklusives Gateway, das auch als Vorgänger den exklusiven else-Schlüssel hat. Auf das zusammenführende exklusive Gateway folgt Aktivität 2, die den ursprünglichen Nachfolger der Aktivität als Nachfolger hat. Durch dieses Konstrukt wird sichergestellt, dass die gesamte Aktivität immer ausgeführt wird, unabhängig davon, ob das angeheftete Ereignis eintritt.

Darstellung der Normalisierungsregel

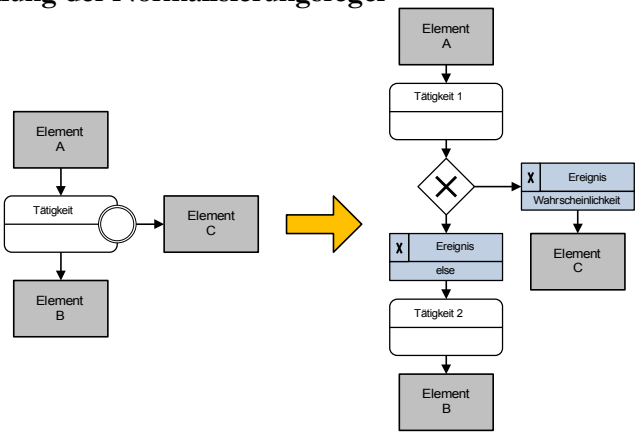


Normalisierungsregel aNR_6
If an activity has an interrupting boundary event, then the activity is divided in two activities. The first activity will get a splitting exclusive gateway as it successor. This exclusive gateway will get two successors, an exclusive key and an exclusive else key. The exclusive key will get the successor of the boundary event as its successor. The else key will get the second activity as successor and this activity will get the successor of the original activity as successor.

Normalisierungsregel aNR_6 überführt ein angeheftetes unterbrechendes Ereignis in einfachere Elemente. Analog zur Normalisierungsregel aNR_5 wird die Aktivität in zwei Aktivitäten unterteilt. Auf die erste Aktivität folgt ein verzweigendes exklusives Gateway, welches das Eintreten des angehefteten Ereignisses während der Bearbeitung prüft. Ein exklusive Schlüssel beinhaltet die Wahrscheinlichkeit für das Eintreten des Ereignisses. Auf dieses folgt der Nachfolger des angehefteten Ereignisses. Als zweiten Nachfolger erhält das exklusive

Gateway einen exklusiven else-Schlüssel, der das Nichteintreten des Ereignisses berücksichtigt. Auf diesen folgt die zweite Tätigkeit, die den ursprünglichen Nachfolger der Aktivität als Nachfolger hat. Dieses Konstrukt prüft nach einer definierten Bearbeitungszeit, ob das angeheftete Ereignis eintritt, womit die Bearbeitung der Aktivität unterbrochen wird.

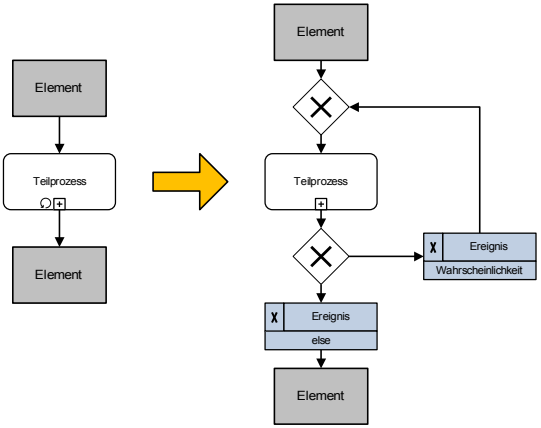
Darstellung der Normalisierungsregel



Normalisierungsregel aNR_7
If a sub process has a loop marker, then the marker is removed and the sub process will get a joining exclusive gateway as its predecessor and a splitting exclusive gateway as its successor. The splitting gateway will get as a first successor an exclusive else key that will get the successor of the sub process as its successor and as a second successor an exclusive key that has the joining exclusive gateway as its successor.

Normalisierungsregel aNR_7 ist das Äquivalent zur Normalisierungsregel aNR_2 für den Teilprozess. Für die Erläuterung wird auf diese Normalisierungsregel verwiesen.

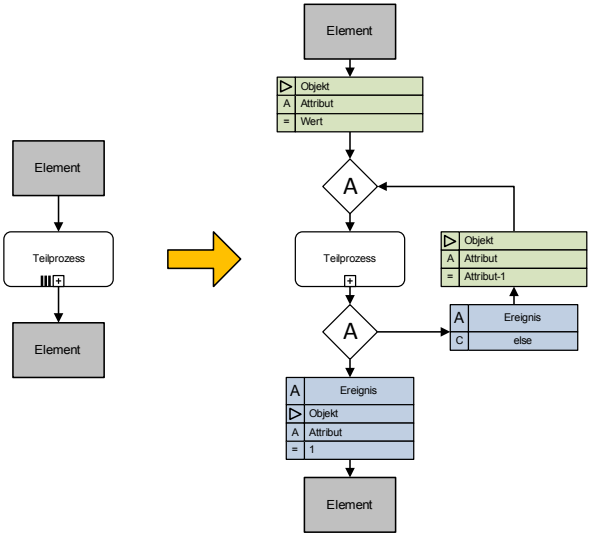
Darstellung der Normalisierungsregel



Normalisierungsregel aNR_8
If a sub process has a sequence multi-instance marker, then the marker is removed and the sub process will get a joining attribute-based gateway as its predecessor and a splitting attribute-based gateway as its successor. The splitting gateway will get as a first successor an attribute-based key that will get the successor of the sub process as its successor and as a second successor an attribute-based else key that will get an additional attribute set as its successor and the attribute set will get the joining exclusive gateway as its successor. The joining gateway will get an attribute set as its predecessor and this element will get the predecessor of the sub process as its predecessor.

Normalisierungsregel aNR_8 wandelt den „Sequence Multi-Instance“ Marker bei einem Teilprozess in einfachere Elemente um. Diese Regel entspricht der Normalisierungsregel aNR_3, welche die Aktivität adressiert. Analog zu dieser Normalisierungsregel werden zusätzliche Elemente mittels des attributsbasierten Gateways hinzugefügt, nur dass anstelle einer Aktivität von einem Teilprozess gesprochen wird.

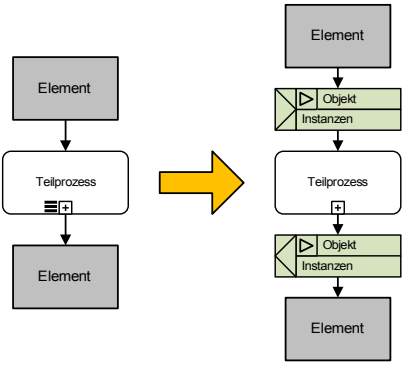
Darstellung der Normalisierungsregel



Normalisierungsregel aNR_9
If a sub process has a sequence multi-instance marker, then the marker is removed and the activity will get a branch instance as predecessor and a merge instance as a successor.

Die Normalisierungsregel aNR_9 adressiert einen Teilprozess, der mit einem „Parallel Multi-Instance“ Marker gekennzeichnet ist. Der Teilprozess wird in diesem Falle mehr als einmal unabhängig voneinander durchgeführt. Die Regel entspricht der Normalisierungsregel aNR_4.

Darstellung der Normalisierungsregel

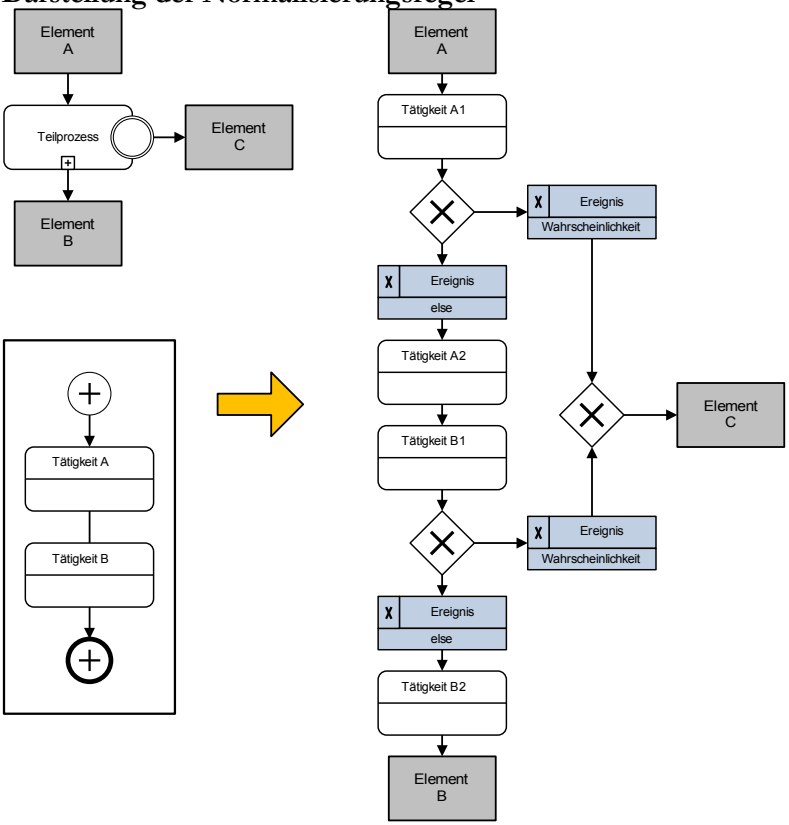


Normalisierungsregel aNR_{10}
If a sub process has an interrupting boundary event, then the sub process will be removed. The connection elements of the sub process will be removed and the remaining elements will be placed between the predecessor and successor of the sub process. The successor of the boundary event will get a combing exclusive gateway as predecessor. Every activity will be divided in two activities. Every first activity will get a splitting exclusive gateway as it successor. This exclusive gateway will get two successors, an exclusive key and an exclusive else key. Every exclusive key will get the combining exclusive gateway of the successor of the boundary event as its successor. The else key will get the second activity as successor and this activity will get the successor of the original activity as successor.

Die Normalisierungsregel aNR_{10} betrachtet einen Teilprozess mit einem angehängten unterbrechenden Ereignis. Da in einigen Simulationsumgebungen ein entsprechendes Element für den Teilprozess nur einen Eingang und einen Ausgang aufweist, wird der Teilprozess durch diese Normalisierungsregel aufgelöst. Der Prozessablauf des Teilprozesses wird mit Ausnahme der Verbindungselemente in das übergeordnete ProSiT Ablaufdiagramm integriert und entsprechend mit dem Vorgänger und Nachfolger des Teilprozesses verknüpft. Da unterstellt wird, dass sich in einem Teilprozess mehr als eine Aktivität befindet, erhält der Nachfolger des angehefteten Ereignisses ein zusammenführendes exklusives Gateway als Vorgänger.

Zur Berücksichtigung des unterbrechenden Ereignisses, wird wie in Normalisierungsregel aNR_6 jede Aktivität aufgeteilt und mittels eines exklusiven und parallelen Gateways mit dem zusammenführenden exklusiven Gateway verknüpft. Für eine detailliertere Erläuterung dieses Teilschrittes wird auf die Normalisierungsregel aNR_6 verwiesen. Das dargestellte Beispiel stellt den einfachen Fall dar, dass der Teilprozess lediglich aus zwei Aktivitäten, sowie zwei Verbindungselementen des Teilprozesses.

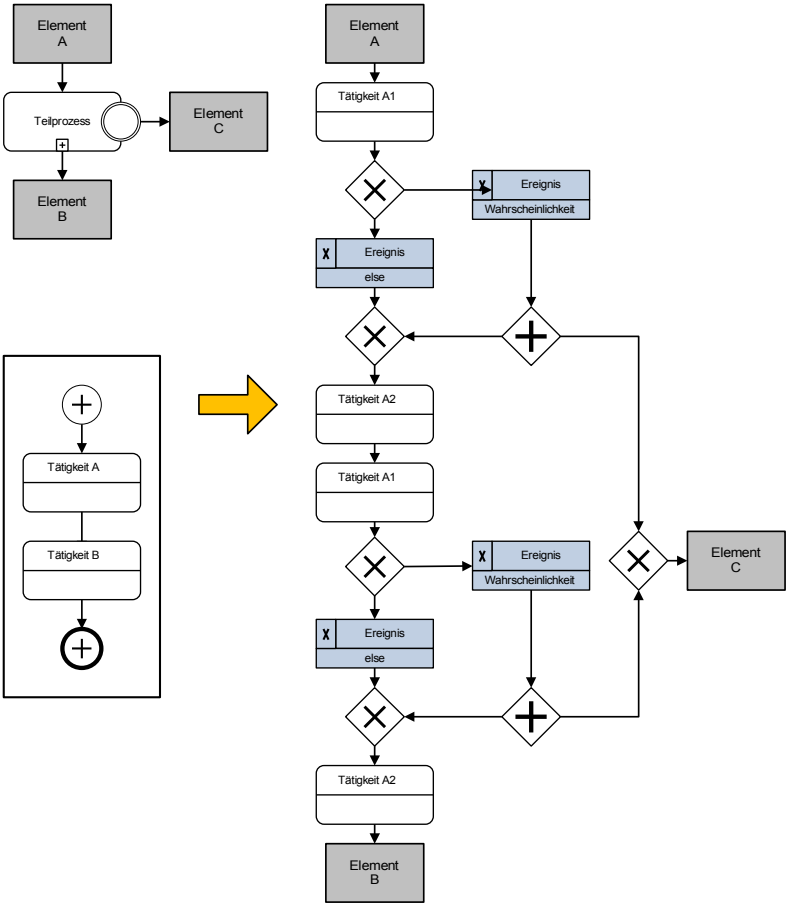
Darstellung der Normalisierungsregel



Normalisierungsregel aNR_{11}
If a sub process has a non-interrupting boundary event, then the sub process will be removed. The connection elements of the sub process will be removed and the remaining elements will be placed between the predecessor and successor of the sub process. The successor of the boundary event will get a combing exclusive gateway as predecessor. Every activity will be divided in two activities. Every first activity will get a splitting exclusive gateway as successor. This exclusive gateway will get two successors, an exclusive key and an exclusive else key. The exclusive else key will get a joining exclusive gateway as successor and the exclusive key will get a splitting parallel gateway as successor. The parallel gateway has as a first successor the combining exclusive gateway of the successor of the boundary event as its successor and as a second successor the joining exclusive gateway. The joining exclusive gateway will get the second activity as successor and the activity will get the original successor of the activity as successor.

Die Normalisierungsregel aNR_{12} entfernt analog zur Normalisierungsregel aNR_{10} ein Teilprozess mit einem angehefteten nicht unterbrechenden Ereignis aus dem ProSiT Ablaufdiagramm. Vom Inhalt des Teilprozesses werden lediglich die Verbindungselemente entfernt und die restlichen Elemente in das umgebende Ablaufdiagramm integriert. Der Nachfolger des angehefteten Ereignisses erhält wie in der Normalisierungsregel aNR_{10} ein zusammenführendes exklusives Ereignis und auf alle Aktivitäten wird die Normalisierungsregel aNR_5 angewendet. Entsprechend wird für die Erläuterung die zur Aufteilung der Aktivitäten führt auf diese Normalisierungsregel verwiesen.

Darstellung der Normalisierungsregel



Linguistische Normalisierungsregeln

Für die Untermenge der linguistischen Normalisierungsregeln muss eine (automatische) Klassifizierung der Wortarten angewendet werden, um diese automatisch auszuführen. Die Konzeption eines entsprechenden Verfahrens ist jedoch nicht Gegenstand dieser Arbeit. Hierzu sind weiterführende Analysen über die Verwendung von Wortarten in Geschäftsprozessmodellen notwendig.

Eine Arbeit, die als Grundlage für die automatische Klassifikation der Wortarten verwendet werden kann, ist beispielsweise von Bassenge (2002). Gegenstand der Dissertation von Bassenge (2002, S. 64-67) ist die automatische Klassifizierung von Wortformen. Hierbei wurde ein Verfahren entwickelt, das isolierte Wörter klassifizieren kann, ohne einen umgebenden Satz dafür zu benötigten.

Verschiedene Artikel beschäftigen sich aber auch mit den Bezeichnungen in Geschäftsprozessmodellen. Diese beschränken sich aber vornehmlich auf die Qualität der Bezeichnungen und zerlegen diese nicht weiter, um daraus Kenntnisse über das Geschäftsprozessmodell zu erlangen. Der Artikel von Friedrich (2009) beschäftigt sich mit der Messung der semantischen Bezeichnungsqualität mittels WordNet. WordNet stellt hierbei eine lexikalische Datenbank, die semantische Relationen von Wörtern beinhaltet (Miller, 1995, S. 40). Diese umfasst unter anderem Synonyme oder Antonyme. Friedrich (2009, S. 8-10) verwendet WordNet zur quantitativen Messung von Qualitätsmetriken. Nach einer Metrik von Friedrich (2009, S. 10-11) kann die Qualität verbessert werden, wenn anstelle von Synonymen der am häufigsten oder standardmäßig verwendete Begriff wird.

Wie Friedrich (2009) verwendeten Mendling und Reijers (2008) das SAP Referenzmodell als Grundlage für die Untersuchung von qualitätsbezogenen Aspekten. Bei Mendling und Reijers (2008, S. 118-119) sind die (englischen) Aktivitäten im Mittelpunkt der Untersuchung. Hierbei

ermittelten die Autoren drei Kategorien von Bezeichnungsvarianten[18]: Verb-Object Stil, Action-Noun Stil und Sonstige Stile. Im SAP Referenzmodell, mit 19.838 Aktivitäten, entsprachen 60% dem Verb-Object Stil, 34% dem Action-Noun Stil und 6% einem sonstigen Stil. Der Verb-Object Stil ist, nach einer Hypothese von Mendling und Reijers (2008, S. 120), der verständlichste Stil zur Benennung von Aktivitäten ist. Bei einer Untersuchung mit 29 Teilnehmern (Studenten) wurde diese Hypothese bestätigt (Mendling et al. 2009). Da die Teilnehmer nur aus der Gruppe der Studenten stammen und es sich lediglich um 29 Teilnehmern handelt, ist die gleiche Kritik wie zum Artikel von Sarshar und Loos (2005) anzubringen, der die Benutzerfreundlichkeit von eEPKs und Petri-Netzen untersucht. Aus diesem Grund kann die Schlussfolgerung, dass im Englischen der Verb-Object Stil am verständlichsten sei, nur als These angesehen werden. Im SAP Referenzmodell wird dieser Stil jedoch am häufigsten verwendet.

Der Stil für die Formulierung von Bezeichnungen ist abhängig von der Sprache. Wenn der Verb-Object Stil im Englischen am häufigsten verwendet wird; stellt sich die Frage, welcher Stil im Deutschen am meisten verwendet wird? Aus dem Artikel von Delfmann et al. (2008, S. 26-28) gehen Stile hervor: „<Substantiv> + <Verb im Infinitiv>“ (Nomen-Verb Kombination) sowie „<Verb im Imperativ> + <Substantiv im Singular>“ (Verb-Nomen Kombination). Das Verb im Infinitiv entspricht den Ausführungen von Gadatsch (2010, S. 191), in dessen Einführungsbuch zum Geschäftsprozessmanagement. Welcher dieser beiden Stile dominierend verwendet wird, lässt sich nicht mit Sicherheit sagen, da hierfür keine referierte Untersuchung vorliegt. Zur Bestimmung einer Tendenz kann aber die studentische Arbeit von Schmidt (2010) herangezogen werden. Ziel der Arbeit war die Untersuchung, welcher Stil in einer EPK wie oft verwendet wird. Gegenstand der Untersuchung waren ausschließlich deutschsprachige Bücher. Als Suchdatenbanken wurden der Gemeinsame Verbund-

18 Beispiele für die Stil-Kategorien sind nach Mendling und Reijers (2008, S. 118-119):
Verb-Object Stil: „Process Cost Planning“
Action-Noun Stil: „Notification Printing“
Sonstige Stile: „Status Analysis Cash Position“

katalog GVK-Plus sowie Google-Books mit den Stichworten „Wirtschaftsinformatik“, „EPK“ und „erweiterte Prozesskette“ verwendet. Von 30 Literaturquellen aus den Jahren 1997 bis 2010 wurden in 26 mindestens einmal die Nomen-Verb Kombination verwendet und in 20 Quellen davon diese Kombination nur ausschließlich. Die Verb-Nomen Kombination wurde im Gegensatz dazu nur in 4 Quellen mindestens einmal verwendet und nur in 2 Quellen ausschließlich. In 8 Quellen wurden aber auch Stile gefunden, die nicht diesen beiden Stilen entsprachen. Untersucht wurde demnach nicht die Anzahl von Prozessmodellen mit einem bestimmten Stil, sondern lediglich das Auftreten in der Literaturquelle selbst. Die Untersuchung selbst müsste auf einer breitere Basis gestellt werden, in dem einzelne Prozessmodelle aufgeschlüsselt werden. Ebenfalls sollten Zeitschriftenaufsätze und Konferenzbeiträge in die Untersuchung eingeschlossen werden, um einen festen empirischen Beleg zu erhalten. Aus der Arbeit kann aber die These abgeleitet werden, dass die Nomen-Verb Kombination im deutschen am häufigsten Verwendung findet. Dieser Stil zur Bezeichnungen von Funktionen und Aktivitäten in Geschäftsprozessmodellen wird im Rahmen der weiteren Arbeit unterstellt.

Zur Bestimmung des Stiles, aber auch zur Bestimmung der verwendeten Wortarten in der Nomen-Verb Kombination sind Verfahren notwendig, welche eine automatische Erkennung ermöglichen. Leopold et al. (2009) untersuchten verschiedene Verfahren deren die Erfolgsquote zur Erkennung der Wortarten. Bezogen auf das SAP Referenzmodell bestehen die Bezeichnungen von etwa 18% der Funktionen aus zwei Wörtern (Leopold et al. 2009, S. 51). Die Funktionen mit zwei bis vier Wörtern in der Bezeichnung machen etwa 70% der Funktionen aus. Diese Verteilung deckt sich mit der Verteilung der Anzahl der Wörter bei Delfman et al. (2008, S. 180-181); im Rahmen eines vier monatlichen Modellierungsprojektes wurden 4805 EPK-Modelle mit 13.935 Funktionen erstellt. Auch bei dieser Modellen bestehen etwa 70% der Funktionen aus zwei bis vier Wörtern.

Mittels eines Part-of-Speech Tagging Verfahrens gelang es Leopold et al. (2009, S. 98) 94% der 19.838 Wörter des SAP Referenzmodells der richtigen Wortart zuzuweisen. Mit dem TreeTagger (Schmidt 1994; Schmidt 1995), den Delfmann et al. (2008, S. 181) verwenden, konnten fast alle Wortarten automatisch klassifiziert werden. Aufbauend auf diesen Ergebnissen wird für die Arbeit unterstellt: Eine automatische Klassifikation von Wortarten in Aktitiväten von Geschäftsprozessmodellen kann ausgeführt werden.

Für das Repository kann der von Friedrich (2009, S. 10-11) verwendete Ansatz mit WordNet herangezogen werden, um ähnliche Worte gleich zu klassifizieren. Hierfür müssen, für das in der Datenbank aufgeführte Wort, weitere Synonyme herangezogen werden, wenn diese in den Prozessmodellen verwendet werden. Eine Automatisierung ist damit realisierbar. Hierfür soll das von Friedrich (2009, S. 11) dargestellte Beispiel mit den zwei Begriffen „order" und „purchase" herangezogen werden[19]. Für ein Gedankenexperiment wird als Annahme unterstellt: Im Repository ist das Wort „order" hinterlegt, nicht jedoch das Wort „purchase". Im Geschäftsprozessmodell soll das Wort „order" nun sieben Mal vorkommen und einmal das Wort „purchase". Das Repository wird daher nach dem Wort „purchase" untersucht, mit dem Ergebnis, dass es nicht vorhanden ist. Im nächsten Schritt wird nun eine Verbindung zu WordNet aufgebaut und geprüft, welche Synonyme für das Wort vorhanden sind. Die gefundenen Synonyme werden mit dem Repository verglichen. Beim Wort „order" wird das Verfahren fündig. Drei weitere Vorgehensweisen sind jetzt denkbar. Im ersten Fall wird der Modellierer auf den Fund aufmerksam gemacht, der dann einen der weiteren zwei Fälle auswählen kann. Im zweiten Fall wird das Wort separat und nicht als Synonym in das Repository übernommen, mit dem Wort „order" als Grundlage. Es besteht so die Möglichkeit, das Wort anders zu klassifizieren. Im dritten Fall wird das Wort „purchase" als Synonym des Wortes „order" hinterlegt. Dadurch muss die die Klassifikation im Repository für jede Domäne nur einmal

19 Beide Wörter „order" und „purchase" sind für dieses Beispiel als Verben aufzufassen und nicht als Nomen.

gepflegt werden. Neben der manuellen Auswahl der Fälle zwei und drei, wie dieses durch Fall eins angenommen wird, ist auch eine automatische Zuordnung denkbar. Über ein Protokoll könnte der Modellierer die Ergebnisse nachträglich prüfen.

Wie Friedrich (2009, S. 19) aufführt, können zwei Probleme bei der Verwendung von WordNet auftreten. Zum einen können Begriffe darin nicht enthalten sein, womit diese automatische Zuordnung nicht erfolgen kann. Diese würde lediglich ein manuelles Eingreifen bedeuten und in keiner Reduktion des Aufwandes. Problematischer ist aber der zweite Sachverhalt: Fachbegriffe werden in verschiedenen Domäne mit unterschiedlich verwendet. Friedrich (2009, S. 19) führt hierfür das Wort „R/3“ auf. Dieses wurde aber nicht als IT System erkannt, sondern als universelle Gaskonstante interpretiert.

Basierend auf diesen Ausführungen wird für die nachfolgenden linguistischen Normalisierungsregeln unterstellt, dass eine automatische Klassifikation der Wortarten erfolgt ist, womit auf diese zurückgegriffen werden kann.

Normalisierungsregel aNR_{12}
Wenn das Verb in der Tätigkeit der Aktivität im Repository aufgeführt wird, dann weise der Aktivität die Aktivitätsart zu.

Normalisierungsregel aNR_{13} ist für die Klassifizierung der Aktivität hinsichtlich ihres Typs in Hauptaktivität und Unterstützungsaktivität konzipiert. Damit diese Normalisierungsregel angewendet werden kann, müssen zwei Bedingungen erfüllt sein. Zum einen muss die Domäne (mNR_1) des Transformationsmodells , zum anderen die Prozessart (mNR_2) festgelegt sein (Kloos et al. 2011, S. 29-30). Sind diese Bedingungen erfüllt, erfolgt eine Zuweisung der Aktivitätsart. Bei der Prüfung des Repositories wird neben der Klassifikationstabelle (Tabelle 18) auch die Synonymtabelle (Tabelle 19) geprüft.

Die Ausprägungen des Aktivitätstyps werden im ProSiT Ablaufdiagramm gemäß Abbildung 13 farblich gekennzeichnet. Eine rötliche Farbe steht für eine Hauptaktivität und eine gräuliche für eine Unterstützungsaktivität. Das Ergebnis der Klassifikation kann anschließend für andere Normalisierungsregeln verwendet werden. Der Aktivitätstyp könnte in eine Simulationsumgebung überführt werden, um darauf Analysen durchzuführen. Dieser Aspekt wird in der aktuellen Konzeption des ProSiT Konzepts aber nicht berücksichtigt.

Die Klassifikation ist bei dieser automatischen Normalisierungsregel nur als Vorschlag zu werten. Die Zuweisung basiert nur auf den Verben. Auch sind die verwendeten Verben abhängig vom Modellierer oder den Modellierungsrichtlinien (Kloos et al. 2010, S. 96). Im semantischen Kontext könnte die Zuweisung daher nicht korrekt sein.

Normalisierungsregel aNR_{13}
Wenn das Objekt in der Tätigkeit der Aktivität dem Prozessobjekt entspricht, dann wird die Aktivität als Ausführung am Prozessobjekt klassifiziert. Ist dies nicht der Fall, dann wird das Verb in der Tätigkeit der Aktivität mit dem Repository abgeglichen und eine entsprechende Zuweisung zur Ausführung am Prozessobjekt umgesetzt.

Normalisierungsregel aNR_{13} klassifiziert eine Aktivität hinsichtlich der Ausführung am Prozessobjekt. Hierfür wird zunächst das Objekt, im grammatikalischen Sinne, aus der Tätigkeit der Aktivität extrahiert. Hierbei handelt es sich in der Regel um den Akkusativ. Abhängig vom Verb kann das Objekt der Tätigkeit aber auch der Genitiv oder Dativ sein. Entspricht das Objekt dem Prozessobjekt, folgt daraus transzendental, dass die Aktivität am Prozessobjekt ausgeführt wird. Trifft dies nicht zu, wird das Verb herangezogen und mit dem Repository abgeglichen. Hierbei sind die gleichen Aspekte zu berücksichtigen, die bei der automatischen Normalisierungsregel aNR_{12} aufgeführt wurden; die Zuweisung ist nur abhängig vom Verb. Im Gegensatz zur Normalisierungsregel aNR_{12} gibt es aber bei der Normalisierungsregel aNR_{13} zwei Varianten um die Ausführung am Prozessobjekt zu klassifizieren.

Normalisierungsregel aNR_{14}
Wenn in der Tätigkeit Genitive verwendet werden, die nicht das Objekt der Tätigkeit sind, dann werden die Genitive aus der Tätigkeit entfernt und als Ressourcen der Aktivität hinzugefügt.

Normalisierungsregel aNR_{14} vollzieht eine sprachliche Analyse der Tätigkeit einer Aktivität. Sie identifiziert Genitive, die nicht das grammatikalische Objekt der Tätigkeit repräsentieren. Ein verbales Beispiel ist der Satz „Der Verkäufer erstellt den Auftrag mit dem ERP-System.“. Der Nominativ dieses Satzes stellt in aktiv formulierten Sätzen immer eine Ressource dar. Dieser wird jedoch nicht in der textlichen Beschreibung der Tätigkeit beziehungsweise bei Prozessmodellierungsnotationen in den Beschreibungen verwendet Die eEPK würde den Verkäufer als Stelle darstellen, die mit einer Funktion verknüpft wäre. BPMN sowie das UML Aktivitätsdiagramm realisieren den Nominativ mittels dem Konstrukt einer Schwimmbahn. Aus sprachlicher Sicht und im Hinblick auf die Ressourcensicht handelt es sich beim Nominativ um die ausführende Ressource. Wird hingegen ein passiver Satz verwendet („Der Auftrag wird vom Verkäufer mit dem ERP-System angelegt“), dann wechseln Nominativ und Dativ die Rollen.

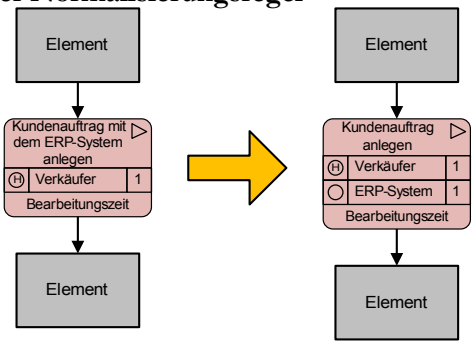
Bei Genitive handelt es sich ebenfalls um Ressourcen, die jedoch unterstützend eingesetzt werden. Im obigen Beispiel wäre dies das ERP-System. Die eEPK kann unterstützende Ressourcen modellieren. In BPMN könnte höchstens mit Artefakten gearbeitet werden. Das UML Aktivitätsdiagramm kann diese aber nicht abbilden; dieses ist für die Modellierung von Aktivitäten konzipiert, die von einem IT-System ausgeführt werden.

Anstelle als separates Element kann der Genitiv auch Teil der verbalen Beschreibungen der Tätigkeiten sein. Die linguistische Normalisierungsregel entfernt diese Genitive aus der Tätigkeit, und fügt sie als Ressource der Aktivität hinzu. Allgemein wird hierbei unterstellt, dass es sich um simulationsrelevante Ressourcen handelt. Wie in Kloos et al.

(2010, S. 96) jedoch aufgeführt wird, bedarf es eine weitere Prüfung, ob diese Ressourcen tatsächlich simulationsrelevant sind. Die Regel ist somit nur als unterstützend aufzufassen.

Die für diese Normalisierungsregel aufgeführten Interpretationen gehen auf eine Darstellung aus dem Buch von Allweyer (2005, S. 130-135) zurück. Allweyer führt verschiedene Darstellungsformen für Prozessmodelle auf, unter anderem eine Beschreibung in Textform und die Beschreibung als semiformales Modell. Basierend auf diesen beiden Darstellungen wurde ein Zusammenhang zwischen der Wortart und der Repräsentation in einer eEPK untersucht und dieser daraufhin verallgemeinert. Diese verwendeten Zusammenhänge bedürfen einer weiterführenden Untersuchung, wenn weitere linguistische Regeln auf diesen aufbauen sollen, und stellen eine offenen Forschungsfrage dar.

Darstellung der Normalisierungsregel



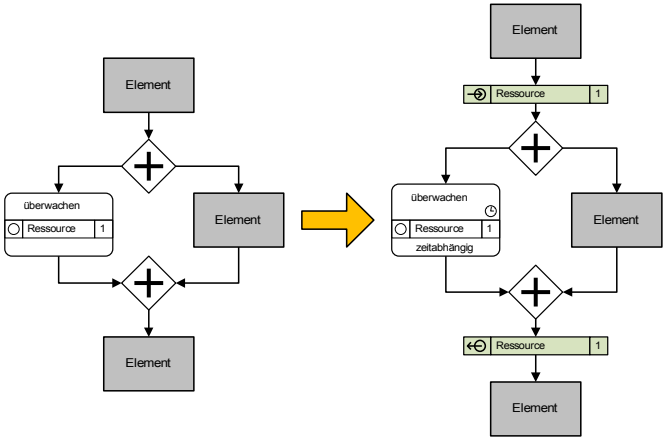
Normalisierungsregel aNR_{15}
Wenn eine Tätigkeit das Verb „überwachen“ beinhaltet und die Aktivität auf ein verzweigendes paralleles Gateway folgt und ein zusammenführendes paralleles Gateway als Nachfolger hat, dann wird vor dem verzweigenden parallelen Gateway eine Ressourcenbindung eingefügt, welche die nicht gebundenen Ressourcen der Aktivität bindet und nach dem zusammenführenden parallelen Gateway eine Ressourcenfreigabe, die diese Ressourcen wieder freigibt. Die Aktivität wird daraufhin mit dem zeitabhängigen Marker versehen.

Die Normalisierungsregel aNR_{15} prüft eine Aktivität auf das Verb „überwachen“. Wird dieses Verb verwendet, kann die Dauer der Überwachung von einer anderen Aktivität abhängen. Im Rahmen der Normalisierungsregel wird dies unterstellt, wenn sich die Aktivität innerhalb eines parallelen Zweiges befindet.

Um dies festzustellen wird geprüft, ob der Vorgänger und Nachfolger der Aktivität jeweils ein paralleles Gateway ist. In diesem Fall wird unterstellt, dass die Dauer der Überwachung von den anderen parallelen Prozesspfaden abhängig ist. Daher wird die Aktivität mit dem Zeitabhängigkeitsmarker versehen. Vor dem verzweigenden parallelen Gateway wird eine Ressourcenbindung eingefügt, die alle nicht gebundenen Ressourcen der Überwachenaktivität beinhaltet. Die Freigabe dieser Ressourcen erfolgt nach dem zusammenführenden parallelen Gateway mit einer entsprechenden Ressourcenfreigabe.

Das Verb „überwachen“ ist für diese Normalisierungsregel als Beispiel aufzufassen. Für welche weiteren Verben dieser Regel angewendet werden könnte, muss über weitere Untersuchungen ermittelt werden. Ein weiterer Untersuchungspunkt kann aus dieser Normalisierungsregel abgeleitet werden. Wenn für das Verb „überwachen“ ein bestimmter Sachverhalt abgeleitet werden kann, gibt es dann andere Verben, für die andere Sachverhalte abgeleitet werden können?

Darstellung der Normalisierungsregel



3.5.2 Semiautomatische Normalisierungsregeln

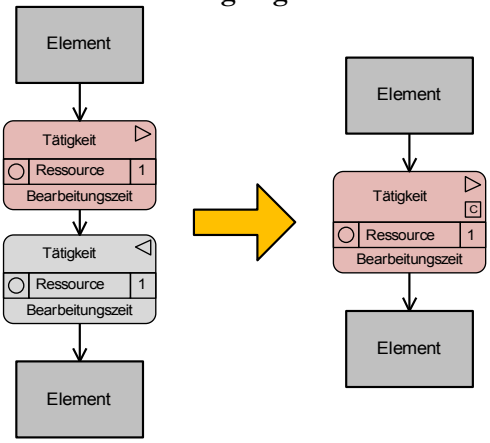
Semiautomatische Normalisierungsregeln werden im Gegensatz zu automatischen Normalisierungsregeln nicht ohne Interaktion mit dem Benutzer durchgeführt. Der Wenn-Teil der Hypothese, mit der Prüfung eines Modellierungskonstrukts, wird automatisch ausgeführt, der Dann-Teil hingegen nur, wenn der Benutzer der Ausführung der Regel zustimmt. Dementsprechend ist in jeder semiautomatischen Normalisierungsregel eine Frage an den Modellierer enthalten, ob die Regel angewendet werden soll.

Normalisierungsregel sNR_1
If the successor of a main activity, which is executed on the process object, is a supporting activity not involving the process object, then query the user, whether the main activity and the subsequent supporting activity can be merged.

Normalisierungsregel sNR_1 identifiziert ein Zweigestirn von Aktivitäten, basierend auf der Klassifikation der Aktivität hinsichtlich der Ausführung am Prozessobjekt. Identifiziert wird eine Hauptaktivität, die am Prozessobjekt ausgeführt wird mit einer darauffolgenden Unterstützungsaktivität, die nicht am Prozessobjekt ausgeführt wird.

Ein Beispiel aus der Ambulanz wäre, wenn auf die Aktivität „Augendruck messen“ die Aktivität „Untersuchungsergebnis drucken“ folgt. Die Normalisierungsregel würde diese beiden Aktivitäten zusammenfassen. Ziel dieser Regel ist die Anzahl der Elemente zu reduzieren. Diese semiautomatische Normalisierungsregel wurde erstmals in Kloos et al. (2010, S. 100) aufgeführt. Im Gegensatz zur Normalisierungsregel sNR_3 werden Ressourcen ignoriert.

Darstellung der Normalisierungsregel

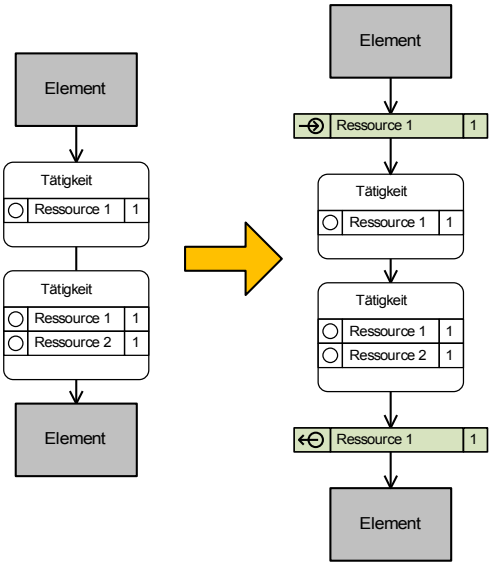


Normalisierungsregel sNR_2
If the same resources are used by sequential activities and the sequential activities have at least one different resource, query the user, whether the resources should be bound for all of these activities.

Die Normalisierungsregel sNR_2 prüft sequenzielle Aktivitäten hinsichtlich gemeinsam verwendeter Ressourcen. Von den Ressourcen muss mindestens eine Aktivität eine weitere Ressource aufweisen. Die Annahme der Normalisierungsregel ist: Zwischen der Ausführung der betroffenen Aktivitäten steht die gemeinsam verwendete Ressource nicht frei zur Verfügung. Dieser Sachverhalt entspricht dem Zahnarzt Gedankenexperiment. Ob dieses Verhalten gewünscht ist, muss der Modellierer bestimmen, da das Verhalten nicht immer unterstellt werden kann.

Diese semiautomatische Normalisierungsregel stellt eine abgewandelte Form der in Kloos et al. (2010, S. 102) aufgeführten Regel dar. Die gemeinsam genutzte Ressource wird durch eine Ressourcenreservierung vor und einer Ressourcefreigabe nach den sequenziellen Aktivitäten gebunden.

Darstellung der Normalisierungsregel

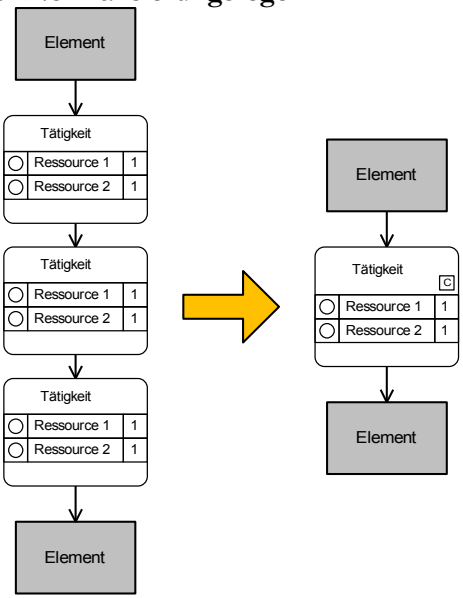


Normalisierungsregel sNR_3
If the same resources are used by sequential activities and all sequential activities have the same resources, query the user, whether the activities can be merged.

Die Normalisierungsregel sNR_3 ist eine Ergänzung zur Regel sNR_2. Bei dieser wird ebenfalls nach gleichen Ressourcen in einer sequenziellen Abfolge von Aktivitäten geprüft. Als Bedingung für die Normalisierungsregel müssen alle Ressourcen in der sequenziellen Abfolge identisch sein. Ist diese Prämisse gegeben, können die einzelnen Aktivitäten aus simulationstechnischer Sicht zusammengefasst werden. Da auf einzelnen Aktivitäten aber Auswertungen durchgeführt werden könnten, muss ein Modellierer diese Regel bestätigen.

Die Tätigkeit der zusammengefassten Aktivität muss vom Modellierer formuliert werden. Entweder wird diese neu formuliert oder eine der zusammengeführten Aktivitäten wird verwendet.

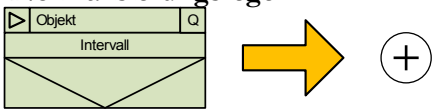
Darstellung der Normalisierungsregel



Normalisierungsregel sNR_4
If a source is placed in a sub process, query the user, whether the source should be converted to a sub process entry point.

Alle Startpunkte von Geschäftsprozessmodellen werden durch Transformationsregeln in Quellen umgewandelt. Dies trifft auch auf Startpunkte zu, die sich in Teilprozessen befinden. Marken werden jedoch nicht von diesen Startpunkten erzeugt, sondern erhalten Marken aus dem umgebenden Prozess. In einem Teilprozess sind dennoch Quellen möglich. Aus diesem Grund muss der Modellierer die einzelnen Quellen prüfen und bestimmen, welche in einen Eintrittspunkt zu umzuwandeln ist. In einem Teilprozess darf nur ein Eintrittspunkt vorhanden sein. Diese Prämissen leitet sich zum einen aus den Implementierungsmöglichkeiten in den Simulationsumgebungen ab, zum anderen aus der Einbindung des Teilprozesses im Prozessablauf. Dieser verfügt nur über einen Vorgänger und einen Nachfolger, womit der Teilprozess nur über einen Eintrittspunkt verfügen kann.

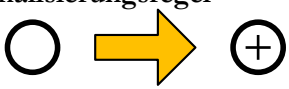
Darstellung der Normalisierungsregel



Normalisierungsregel sNR_5
If a sink is placed in a sub process, query the user, whether the source should be converted to a sub process exit point.

Normalisierungsregel sNR_4 stellt das Gegenstück zu sNR_5 dar. Für diese Regel gelten die gleichen Bedingungen. Anstelle einer Senke wird aber eine Senke in einen Austrittspunkt umgewandelt.

Darstellung der Normalisierungsregel



Normalisierungsregel sNR_6
Wenn eine Aktivität nicht nach der Aktivitätsart klassifiziert ist und das Verb in der Tätigkeit der Aktivität nicht im Repository aufgeführt wird, dann frage den Benutzer, wie die Aktivitätsart zu klassifizieren ist.

Die automatische Normalisierungsregel aNR_{13} fordert als Prämisse ein im Repository definiertes Verb, um die Aktivitätsart zu klassifizieren. Hier greift die semiautomatische Normalisierungsregel sNR_6. Ist das Verb nicht im Repository, dann wird der Benutzer befragt, wie die Aktivitätsart zu klassifizieren ist. Als Erweiterung für die Regel könnte jedes Verb in das Repository aufgenommen werden, beziehungsweise ein Synonym für das Verb angegeben wird. Durch die Anwendung der Normalisierungsregeln aNR_{13} und sNR_6 ist sichergestellt, dass alle Aktivitäten nach ihrem Aktivitätstyp klassifiziert sind.

Normalisierungsregel sNR_7
Wenn eine Aktivität nicht hinsichtlich der Ausführung am Prozessobjekt klassifiziert ist und das Verb in der Tätigkeit der Aktivität nicht im Repository aufgeführt wird, dann frage den Benutzer, wie die Ausführung am Prozessobjekt zu klassifizieren ist.

Analog zu Normalisierungsregel sNR_6 sorgt die Normalisierungsregel sNR_7 für eine Klassifikation der Aktivität hinsichtlich der Ausführung am Prozessobjekt, wenn diese noch nicht klassifiziert ist. Dies ist nach der Anwendung der Normalisierungsregeln aNR_{14} der Fall, wenn das Objekt der Tätigkeit nicht mit dem Prozessobjekt übereinstimmt und das Verb nicht im Repository aufgeführt wird. In diesem Falle wird der Benutzer befragt, ob die Aktivität am Prozessobjekt ausgeführt wird oder ob dies nicht der Fall ist. Wie für die Normalisierungsregel sNR_6 gilt, dass auch ein Synonym oder der Eintrag für das Verb im Repository gepflegt werden kann.

Normalisierungsregel sNR_8
If an activity has a parallel gateway with more than one successor as its predecessor and a parallel gateway with more than one predecessor as its successor and is classified as supporting activity, query the user, if the activity should be converted to a time-dependent activity. If the activity is converted, then a resource binding is added before the predecessor gateway and a resource release is added after the successor gateway for all unbound resources of the activity.

Die Normalisierungsregel sNR_8 kann auf Zustimmung des Modellierers, ähnlich der automatischen Normalisierungsregel aNR_{15}, eine Aktivität in eine zeitabhängige Aktivität überführen. Eine Aktivität muss hierfür ein verzweigendes paralleles Gateway als Vorgänger, ein zusammenführendes Gateway als Nachfolger und als Unterstützungsaktivität klassifiziert sein. Die Bedingungen sind identisch mit der Normalisierungsregel aNR_{15}, allerdings ohne sprachlicher Prüfung.

Bei der Aktivierung der Regel wird die Aktivität in eine zeitabhängige Aktivität überführt. Damit die Ressourcen für die abhängige Zeit reserviert werden, wird vor dem verzweigenden parallelen Gateway eine Ressourcenbindung und nach dem zusammenführenden parallelen

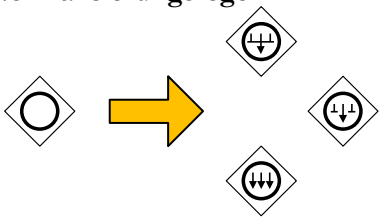
Gateway eine Ressourcenfreigabe für alle nicht gebundenen Ressourcen eingefügt. Durch die Regeln aNR_{15} und sNR_{8} werden zeitabhängige Aktivitäten korrekt umgesetzt. Wird manuell eine Aktivität in eine zeitabhängige Aktivität überführt, dann wird diese im Rahmen eines Simulationsmodells als eine Aktivität mit Bearbeitungszeit von 0 umgesetzt, wodurch es zu falschen Simulationsergebnissen durch freie Ressourcen kommen kann.

Normalisierungsregel sNR_9
If an inclusive gateway has not a marker, query the user, which behavior the gateway represents.

Rittgen (1999, S. 3-4) die drei Fälle „wait-for-all“, „first-come“ und „every-time“ lediglich für zusammenführende Oder Operatoren. Im Rahmen des ProSiT Ablaufdiagramm wird werden diese Fälle aber auch für verzweigende Order Operatoren eingesetzt.

Dies wird bedingt durch die Transformationsregeln zu den Simulationsumgebungen, damit ein inklusives Oder korrekt abgebildet werden kann. Der verzweigende und zusammenführende Operator muss vom gleichen Typ sein, damit das korrekte Verhalten beim zusammenführenden Oder eingehalten wird. Hierfür ist ein entsprechendes Verhalten im verzweigenden Operator notwendig. Unabhängig von verzweigend oder zusammenführend erfolgt die Klassifikation durch die Normalisierungsregel sNR_9.

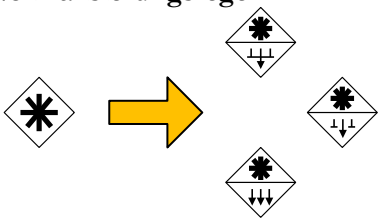
Darstellung der Normalisierungsregel



Normalisierungsregel sNR_{10}
If a complex gateway has not a marker, query the user, which behavior the gateway represents.

Analog zur Normalisierungsregel sNR_9 erfolgt bei der Regel sNR_{10} die gleiche Normalisierung am komplexen Gateway.

Darstellung der Normalisierungsregel



Normalisierungsregel sNR_{11}
If an attribute-based gateway has one predecessor and more than one successor and the gateway has not an attribute-based else key as successor, query the user, which attribute-based key should be converted to an attribute-based else key.

Normalisierungsregel sNR_{11} prüft ein verzweigendes attributsbasiertes Gateway, ob ein attributsbasierter else-Schlüssel vorhanden ist. Das ProSiT Konzept hat keinen Mechanismus implementiert, um sicherzustellen, dass sich attributsbasierte exklusive Entscheidungen einander ausschließen. Um dies sicherzustellen, muss ein Pfad mittels eines else-Schlüssels ausgewählt werden. Mit dieser Maßnahme wird mindestens ein Prozesspfad bei einer Prüfung ausgeführt. Welcher Prozesspfad den else-Schlüssel erhält, muss durch den Modellierer bestimmt werden.

Normalisierungsregel sNR_{12}
If a complex gateway has one predecessor and more than one successor and the gateway has not a complex else key as successor, query the user, which complex key should be converted to a complex else key.

Die Normalisierungsregel sNR_{12} arbeitet, analog zur Normalisierungsregel sNR_{11}, für ein verzweigendes komplexes Gateway. Wenn dieses Gateway keinen komplexen else-Schlüssel aufweist, dann wird der Modellierer durch befragt, welcher Schlüssel umgewandelt werden soll. Diese Umwandlung ist notwendig, damit sicher gestellt wird, um nach dem verzweigenden Gateway immer ein Prozesspfad auszuführen.

Normalisierungsregel sNR_{13}
Wenn das Verb der Aktivität im Repository aufgeführt wird, dann biete dem Modellierer einen entsprechenden Erfassungsbogen für die Erfassung der Bearbeitungszeit an.

Basierend auf den Ausführungen in Kloos et al. (2010, S. 98-99) sowie Kloos et al. (2011, S. 29-31) betrachtet die Normalisierungsregel sNR_{13} die Bearbeitungszeiten von Aktivitäten. Während die eigentliche Erfassung der Bearbeitungszeiten eine manuelle Arbeit ist, wird die Vorbereitung der Erfassung vom ProSiT Konzept unterstützt.

In der Tätigkeitssicht wurde auf das Repository eingegangen. Unter anderem wurde auf einen Vorschlag für die Erfassungsart eingegangen. Basierend auf dieser werden dem Modellierer Erfassungsbögen vorbereitet, die für die Ermittlung der Bearbeitungszeiten verwendet werden können. Dies ist Gegenstand der Normalisierungsregel sNR_{13}. Wenn ein Verb aber nicht im Repository aufgeführt wird, dann wird als für die entsprechende Aktivität als Erfassungsart messen vorgeschlagen. Die konkrete Bearbeitungszeit, inklusive der Verteilung ist jedoch vom Modellierer festzulegen.

3.5.3 Manuelle Normalisierung

Die manuelle Normalisierung beschreibt gemäß der Bezeichnung manuelle Eingriffe des Benutzers. Dies kann zusätzliche Informationen aber auch eine Abstraktion des Modells umfassen. Diese Art des Eingriffs kann nicht in Form von Regeln formuliert werden, da sie abhängig von dem zugrunde gelegten Modell sind.

Statt Regeln werden für die manuelle Normalisierung Anweisungen formuliert, die sich an Normalisierungsregeln orientieren. Einige manuelle Normalisierungen müssen immer angewendet werden, damit das Modell über die notwendigen Informationen verfügt, um simuliert werden zu können.

Nach der Formulierung der Handlungsanweisung folgt eine Beschreibung des Normalisierungsschrittes. Im Gegensatz zum Rest der Regelbasis erfolgt jedoch nicht bei jeder Handlungsanweisung eine grafische Darstellung. Diese wird nur dargestellt, wenn eine sichtbare Änderung im Modell erfolgt.

Die aufgeführten manuellen Normalisierungen sind nicht als umfassend aufzufassen. Basierend auf weiterer Forschungstätigkeit und praktischen Erfahrungen können weitere manuelle Normalisierungsschritte konzipiert werden.

Manuelle Normalisierung mNR_1
Define the Domain of the transformation model.

Die manuelle Normalisierung mNR_1 fordert den Modellierer auf, eine Domäne für das erzeugte Transformationsmodell festzulegen. Die Domäne ist gegenwärtig ausschließlich für linguistische Normalisierungsregel notwendig, welche auf das Repository der Tätigkeitssicht zurückgreifen. Da das gesamte Repository domänenabhängig ist, muss für das Transformationsmodell die Domäne festgelegt werden. Die

Domäne selbst ist keine Eigenschaft eines Geschäftsprozessmodells. Lediglich für die Klassifikation mittels Normalisierungsregeln wird diese benötigt.

Manuelle Normalisierung mNR_2
Define the process type for each flowchart.

Die manuelle Normalisierung mNR_2 fordert den Modellierer auf, den Prozesstyp für jedes ProSiT Ablaufdiagramm festzulegen. Diese Eigenschaft, die gemäß der Definition einem Geschäftsprozess zugesprochen wird, verfügt über die drei Ausprägungen: Führungsprozess, Kernprozess und Unterstützungsprozess. Diese Klassifikation eines Geschäftsprozesses wird ebenfalls, wie die Domäne, für die Verwendung des Repositorys benötigt.

Manuelle Normalisierung mNR_3
Define the process object for each flowchart.

Die manuelle Normalisierung mNR_3 leitet die Definition des Prozessobjekts für jedes ProSiT Ablaufdiagramm an. Die Klassifizierung ist notwendig, damit linguistische Normalisierungsregeln angewendet werden können und damit die Simulationsumgebungen auf die Attribute der Marken zugreifen können. Entsprechend wird das Prozessobjekt bei den Transformationsregeln zu den Simulationsumgebungen berücksichtigt.

Manuelle Normalisierung mNR_4
Define additional resources for the activities, if they are required.

Die manuelle Normalisierung mNR_4 fordert den Modellierer auf, weitere Ressourcen für die Aktivitäten zu definieren, so weit diese für einen Simulationslauf benötigt werden. Dies trifft beispielhaft auf stationäre Ressourcen zu, die nicht im Geschäftsprozessmodell definiert sind.

Manuelle Normalisierung mNR_5
Check if resources can be bound.

Aufbauend auf der manuellen Normalisierung mNR_4 fragt die Normalisierungsregel mNR_5, ob Ressourcen gebunden werden können. Beispielhaft könnte dies für eine stationäre Ressource gültig sein, die für die Ausführung des gesamten Prozesses notwendig ist. Diese Normalisierung entspricht weitestgehend der semiautomatischen Normalisierungsregel sNR_2. Diese kann eine Ressourcenbindung jedoch nur bei sequenziellen Abläufen erkennen.

Manuelle Normalisierung mNR_6
Check if additional elements should be added to the process.

Die manuelle Normalisierung mNR_6 fordert den Modellierer auf, das ProSiT Ablaufdiagramm dahin gehend zu prüfen, ob weitere Elemente in den Prozess aufgenommen werden sollen. Beispielsweise könnte dies eine Verzögerung sein, die im Geschäftsprozessmodell nicht enthalten war. Da aus Sicht des ProSiT Konzepts unklar ist, welche zusätzlichen Elemente dem Modell hinzufügen werden könnten, ist dieser Schritt als manuelle Normalisierung aufzufassen. Diese Normalisierung erlaubt, weitere simulationsrelevante Arbeitsschritte dem Modell hinzuzufügen. Ein Beispiel hierfür zeigt Allweyer (2005, S. 247). Es wird eine Quelle, Aktivität und Senke hinzugefügt, um die anderweitige Auslastung einer Ressource abzubilden.

Manuelle Normalisierung mNR_7
Define the processing time for the activities.

Die manuelle Normalisierung mNR_7 arbeitet mit Ergebnissen, die durch die semiautomatische Normalisierungsregel sNR13 hervorgebracht wurden. Die mit Unterstützung dieser Normalisierungsregel erhobenen Bearbeitungszeiten müssen in das ProSiT Ablaufdiagramm übernommen werden. Dies stellt einen manuellen Arbeitsschritt dar, womit dieser Teil der Arbeit der manuellen Normalisierung zugeordnet wird.

Manuelle Normalisierung mNR_8
Define the processing time for the delay element.

Im Gegensatz zur manuellen Normalisierung mNR_8 bietet das ProSiT Konzept keine Unterstützung zur Zeiterfassung von Verzögerungen. Die Normalisierung mNR_8 fordert den Modellierer daher lediglich auf, die Verzögerungszeit mit entsprechender Verteilung anzugeben.

Manuelle Normalisierung mNR_9
Define the characteristics of the sources.

Neben den zeitverbrauchenden Elementen, der Aktivität und der Verzögerung, ist die Quelle im Rahmen der manuellen Normalisierung mNR_9 zu definieren. Das Objekt der Quelle wird direkt aus dem Prozessobjekt entnommen. Der zeitliche Abstand zwischen erzeugten Marken sowie die Menge zeitgleich erzeugter Marken wird durch die manuelle Normalisierung mNR_9 abgedeckt.

Manuelle Normalisierung mNR_{10}
Define the probabilities for the exclusive keys.

Die manuelle Normalisierung mNR_{10} sieht die Spezifikation der Wahrscheinlichkeiten für exklusive verzweigende Gateways. Diese werden mittels der exklusiven Schlüssel spezifiziert. Die zu einem exklusiven Gateway gehörenden Schlüssel müssen in der Summe eine Wahrscheinlichkeit von 100% ausweisen.

Manuelle Normalisierung mNR_{11}
Define flow paths of the inclusive keys and specify the probability table.

Aufgrund der Realisierung des verzweigenden inklusiven Gateways müssen in den inklusiven Schlüsseln die Pfade definiert sowie eine Wahrscheinlichkeitstabelle definiert werden. Zur Ausführung dieses Schrittes ist die manuelle Normalisierung mNR_{11} konzipiert. Da mittels der Wahrscheinlichkeitstabelle die inklusive Entscheidung in eine exklusive Entscheidung überführt wird, muss die Summe der Wahrscheinlichkeit aller Möglichkeiten 100% ergeben.

Manuelle Normalisierung mNR_{12}
Define the attribute for the attribute based keys.

Analog zur manuellen Normalisierung mNR10 erfolgt in der manuellen Normalisierung mNR_{12} die Spezifikation der attributsbasierten Schlüssel. Bei diesen ist zu beachten, dass stets das gleiche Attribut verwendet wird. Nur damit kann im Rahmen des ProSiT Konzepts eine exklusive Entscheidung sichergestellt werden.

Manuelle Normalisierung mNR_{13}
Define the conditions for the condition based keys.

Ähnlich zur manuellen Normalisierung mNR_{11} sieht die manuelle Normalisierung mNR_{13} die Spezifikation von Schlüsseln vor. Die komplexen Schlüssel werden entsprechend mit Konditionen belegt. Die geprüften Attribute müssen sich im Gegensatz zu attributsbasieren Schlüsseln nicht gegenseitig ausschließen, sodass eine inklusive Entscheidung realisiert werden kann.

Manuelle Normalisierung mNR_{14}
Define the resources in the resource view.

Neben der Definition der Ressourcen im ProSiT Ablaufdiagramm, die mittels der manuellen Normalisierung mNR_4 gepflegt werden, müssen die Ressourcen in der Ressourcensicht spezifiziert werden. Zum einen müssen die Ressourcen einem Typ zugeordnet, zum anderen die verfügbare Anzahl spezifiziert werden.

Manuelle Normalisierung mNR_{15}
Define the schedule in the resource view.

Der Ressourcensicht ist ebenfalls der Schichtplan zugeordnet. Wenn bei einer Ressource ein Schichtplan verwendet wird, dann muss dieser mit Details versehen werden. Diese Anreicherung ist Gegenstand der manuellen Normalisierung mNR_{15}.

Manuelle Normalisierung mNR_{16}
Define the objects in the object view.

Gegenstand der manuellen Normalisierung mNR_{17} ist die Spezifikation der Objekte der Objektsicht. Die manuellen Tätigkeiten der Regel umfassen die Klassifikation der Attribute, damit diese durch entsprechende Transformationsregeln überführt werden können.

3.6 Die Überführung des Transformationsmodells in die Zielumgebungen

Dieses Kapitel beinhaltet die Transformationsregeln zu den Simulationsumgebungen. Die Regeln sind durch ein induktives Verfahren entstanden. Basierend auf dem semantischen Verhalten der Elemente im Transformationsmodell wurden äquivalente Elemente in den Simulationsumgebungen identifiziert.

Ein Element aus dem Transformationsmodell kann zumeist mit einem Element in der Simulationsumgebung abgebildet werden. Es sind aber auch Elemente vorhanden, die über ein umfassenderes Konstrukt in der Simulationsumgebung abgebildet werden müssen. Ausgangspunkt ist aber immer ein Element im Transformationsmodell.

3.6.1 Vom Transformationsmodell zu AnyLogic

In diesem Kapitel werden die Anwendbarkeits- und Transformationsregeln nach AnyLogic aufgeführt. Mittels der Transformationsregeln können die im ProSiT Ablaufdiagramm definierten Elemente nach AnyLogic überführt werden. Die dazugehörigen Anwendbarkeitsregeln werden zuvor nachfolgend erläutert.

Transformationsregel $alTR_{16}$ in Kombination mit $alTR_{17}$, sollte bei der Überführung vermieden werden. Mit diesen werden zusammengehörende Instanziierungselemente nach AnyLogic überführt. Aufgrund der Realisierung der Transformationsregel $alTR_{17}$ müssen alle Elemente die sich zwischen einer verzweigenden und zusammenführenden Instanziierung befinden für jede auszuführende Instanz dupliziert werden, woran die Übersichtlichkeit des Simulationsmodell leidet. Aufgrund dessen enthält die Anwendbarkeitsregel $alAR_1$ eine sollte Formulierung, ein ProSiT Ablaufdiagramm mit einer zusammenführenden Instanziierung nicht zu überführen.

Anwendbarkeitsregel $alAR_1$
If the process contains a combining instance, then the process should not be converted to AnyLogic.

Strenger ist die Anwendbarkeitsregel für vier andere Elemente des ProSiT Ablaufdiagramms formuliert. Die Transformationsregeln für ein verzweigendes inklusives oder komplexes first-come und every-time Gateway, $alTR_{26}$, $alTR_{27}$, $alTR_{32}$ und $alTR_{33}$, dürfen nicht angewendet werden, wenn eine Marke eine gebundene Ressource aufweist.

Anwendbarkeitsregel $alAR_2$
If an entity has a bound resource, when it arrives at an inclusive first-come or every-time gateway with more than one successor or a complex first-come or every-time gateway with more than one successor, then the process can not be converted to AnyLogic.

Der Grund für die Anwendbarkeitsregel $alAR_2$ liegt am Ressourcensystem der Simulationsumgebung. Bei dieser wird durch ein Seize Element eine Ressource direkt an eine Marke angefügt. Diese Marke kann nur aus dem Simulationsmodell entfernt werden, wenn alle Ressourcen vor dem Erreichen bei einem Sink Element freigegeben werden. Andernfalls führt dies bei einem Simulationslauf zu einem Laufzeitfehler, womit eine Fortführung der Simulation nicht möglich ist. Dieser Fall tritt ein, wenn die aufgeführten Gateways nach AnyLogic überführt werden. Da bei einer Kopie die Ressourcen nicht an die Kopie übergeben werden, können Marken mit gebundenen Ressourcen das Sink Element erreichen. Bezogen auf die Transformationsregel $alTR_{26,}$ tritt dieser Fall ein, wenn der erste Prozesspfad nicht ausgeführt werden soll, womit das Original aus dem Simulationsmodell entfernt wird. Sind an die Marke aber keine Ressourcen gebunden, dann kann das Ergebnis der vier Transformationsregeln fehlerfrei verwendet werden.

Die dritte Anwendbarkeitsregel $alAR_3$ bezieht sich auf die terminierende Senke. Da in AnyLogic keine globale Steuerung der Marken erfolgt und erzeugte Kopien nicht mit dem Original in

Verbindung stehen, ist die terminierende Senke nicht umsetzbar. Wenn demnach ein ProSiT Ablaufdiagramm eine terminierende Senke enthält, kann das Modell nicht überführt werden.

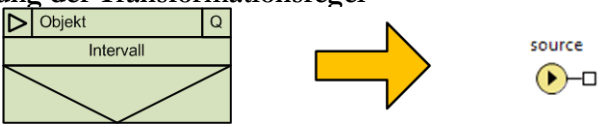
Anwendbarkeitsregel alAR$_3$
If the process contains terminating sink, then the process can not be converted to AnyLogic.

Transformationsregel – alTR (AnyLogic Transformation Rule)
Mittels der nachfolgenden Transformationsregeln erfolgt eine Überführung des Transformationsmodells in die Simulationsumgebung AnyLogic.

Transformationsregel alTR$_1$
If the element is a source, then it will be a source element.

Transformationsregel alTR$_1$ überführt die Quelle in das gleichnamige Element in AnyLogic, das Source Element. Beide Elemente erzeugen Marken, die das Simulationsmodell durchlaufen. Die Quantität der Quelle wird in das Attribute „Entities per arrival“ des Source Elements überführt und der Intervall in das Attribute „Arrival rate“. Zum Hinterlegen des Intervalls ist es jedoch notwendig, das Attribut „Arrivals defined by“ mit dem Wert „Interarrival time“ zu versehen. Damit im Simulationsmodell auf die Elemente des Objekts zurückgegriffen werden kann, muss darüber hinaus das Objekt als „New entity“ verwendet werden.

Darstellung der Transformationsregel



Transformationsregel $alTR_2$
If the element is a sink, then it will be a sink element.

Die Transformationsregel $alTR_2$ überführt die Senke in das Sink Element von AnyLogic. Beide Elemente entfernen eine Marke aus dem Simulationsmodell. Für die Auswertung der Simulationsergebnisse muss das Attribute „Entity class“ mit dem Prozessobjekt versehen.

Darstellung der Transformationsregel

Transformationsregel $alTR_3$
If an activity has not a marker or has a combined Marker, then it will be a service element. The processing time of the activity will be converted to the service element and if the activity has one unbound resource it will be added to the service element and if the activity has more than one resource that is not bound, then for every resource a seize and release element will be added before and after the service element.

Transformationsregel $alTR_3$ ist zweigeteilt hinsichtlich der Anzahl nicht gebundenen Ressourcen. Allgemein wird das semantische Verhalten einer Aktivität durch das Service Element abgedeckt. Dieses bindet eine Ressource, verzögert die Marke und gibt anschließend die gebundene Ressource wieder frei. Diese Element hat aber die Einschränkung: Es kann nur eine Ressource gebunden werden. Bei nur einer gebundenen Ressource kann das Service Element verwendet werden. Hierfür wird die „Entity class“ auf das Prozessobjekt eingestellt und die Bearbeitungszeit in das Attribute „Delay time“ überführt. Über das Attribute „ResourcePool object“ wird die zu bindende Ressource angegeben und über das Attribute „Resource quantity“ die notwendige Anzahl der Ressource. Das Attribute „Maximum queue capacity“ wird letztlich noch ausgewählt, da die Aktivität die Warteschlange nicht einschränkt. Dieser Fall ist in der nachfolgenden Abbildung visualisiert.

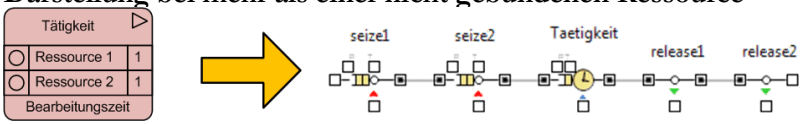
Darstellung bei einer nicht gebundene Ressource

Weist die Aktivität mehr als eine nicht gebundene Ressource auf, dann ist eine Realisierung mit einem Service Element nicht ausreichend. Es sind zusätzliche Seize und Release Elemente für jede nicht gebundene Ressource notwendig.

Analog zu den Ausführungen bei einer nicht gebundenen Ressource wird die Bearbeitungszeit und das Prozessobjekt in das Service Element überführt. In den Seize Elementen wird, basierend auf den nicht gebundenen Ressourcen der Aktivität, das Attribute „Resource quantity“ und „ResourcePool object“ gepflegt. Im entsprechenden Release Element wird die negative Anzahl der benötigten Ressourcen in das Attribute „Quantity released“ eingetragen sowie das „ResourcePool object“. Da eine Aktivität keine Warteschlangenkapazitäten definiert, ist im Service Elemente das Attribute „Maximum queue capacity“ auszuwählen.

Neben dieser Realisierung wäre noch die Kombination aus beiden aufgeführten Fällen möglich. Damit jedoch die Ressourcen einer Aktivität gleichrangig behandelt werden, wird diese Möglichkeit nicht verfolgt.

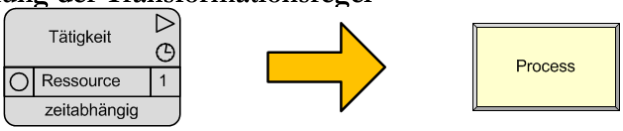
Darstellung bei mehr als einer nicht gebundenen Ressource



Transformationsregel alTR$_4$
If an activity has a time dependent marker, then it will be a service element without a processing time.

Die Transformationsregel alTR$_4$ überführt eine zeitabhängige Tätigkeit nach AnyLogic. Gemäß der Transformationsregel alTR$_3$ wird eine Aktivität mit einem Service Element beschrieben. Eine zeitabhängige Aktivität verfügt aber über keine selbst bestimmbare Bearbeitungszeit. Das Attribut „Delay time“ wird hierfür mit dem Wert 0 versehen und die „Entity class“ auf das Prozessobjekt eingestellt. Aufgrund der automatischen Normalisierungsregel aNR$_{16}$ und der semiautomatischen Normalisierungsregel sNR$_8$ werden die Ressourcen über andere Elemente gebunden. Eine Bindung der Ressourcen im Service Elements oder analog des zweiten Falls der Transformationsregel alTR$_3$ ist nicht sinnvoll. Aufgrund der fehlenden Bearbeitungszeit würden die Ressourcen sofort wieder freigegeben.

Darstellung der Transformationsregel

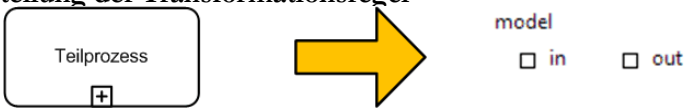


Transformationsregel alTR$_5$
If the element is a sub process, then it will be a model element.

Transformationsregel alTR$_5$ überführt einen Teilprozess in ein separates Model Element in AnyLogic. Alle Elemente des Teilprozesses werden in diesem separaten Modell Element erzeugt. In AnyLogic wird hierfür eine aktive Objektklasse erzeugt. Eine grafische Repräsentation für ein Model Element gibt es in AnyLogic nicht. Eine grafische Repräsentation ergibt sich erst, wenn das Element in einem anderen Modell eingebunden wird. Zusätzlich müssen in diesen Modell Element die Transformationsregeln alTR$_{12}$ und alTR$_{13}$ angewendet werden. Beide Transformationsregeln erzeugen einen Port für eingehende und für ausgehende Marken. Mit diesen Marken kann das Element im im

Ablauf eingebunden werden. Exemplarisch ist das Ergebnis dieser drei Transformationsregel nachfolgend dargestellt.

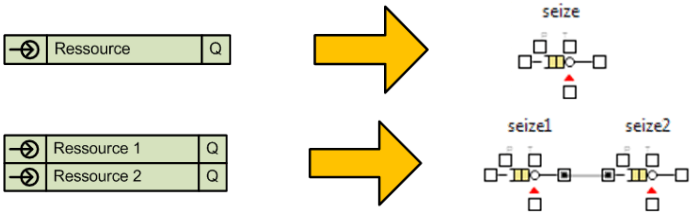
Darstellung der Transformationsregel



Transformationsregel alTR$_6$
If the element is a resource binding, then it will be a series of seize element for each resource.

Transformationsregel alTR$_6$ überführt eine Ressourcenbindung in ein Seize Element. Während im ProSiT Ablaufdiagramm mehrere Ressourcen über eine Ressourcenbindung reserviert werden können, kann ein Seize Element in AnyLogic nur eine Ressource binden. Werden mehrere Ressourcen verwendet, ist eine sequentielle Abfolge von Seize Elemente für die einzelnen Ressourcen notwendig. In diese Elemente wird Anzahl der zu bindenden Ressourcen in das Attribute „Resource quantity“ überführt. Der Name der Ressource wird im Attribut „ResourcePool object“ hinterlegt. Wie die Aktivität definiert das Seize Element keine Beschränkung der Kapazität, womit ebenfalls das Attribute „Maximum queue capacity“ ausgewählt wird.

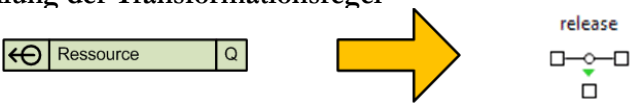
Darstellung der Transformationsregel



Transformationsregel $alTR_7$
If the element is a resource release, then it will be a series of release element for each resource.

Transformationsregel $alTR_7$ ist das simulationstechnische Gegenstück zur Transformationsregel $alTR_6$. Mit dieser Regel erfolgt die Überführung der Ressourcenfreigabe in Release Elemente. Wie beim Seize Elemente können nur Ressourcen eines Typs freigegeben werden. Bei mehreren Ressourcen ist eine sequenzielle Abfolge von Release Elementen notwendig. Der Name der Ressource wird in das Attribut „ResourcePool object“ überführt und die freizugebende Quantität Q wird negiert $-Q$ in das Attribut „Quantity released“ überführt.

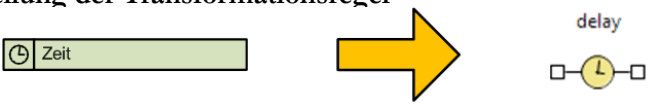
Darstellung der Transformationsregel



Transformationsregel $alTR_8$
If the element is a delay, then it will be a delay element.

Transformationsregel $alTR_8$ überführt eine Verzögerung in das gleichnamige Delay Element. Beide Elemente verzögern eine Marke um die angegebene Zeitdauer. Diese Zeitdauer wird in das Attribut „Delay time“ überführt. Wird eine Marke verzögert, dann ist diese unabhängig von anderen Marken. Physische Begrenzungen, wie diese bei einem Simulationsmodell eines Produktionssystems auftreten, werden bei Geschäftsprozesssimulationen in der Regel nicht berücksichtigt. Daher wird die Kapazität der Verzögerung auf unbegrenzt gesetzt, in dem das Attribut „Maximum capacity“ ausgewählt wird.

Darstellung der Transformationsregel



Transformationsregel $alTR_9$
If the element is an attribute set, then it will be a delay element with an OnExit method.

Transformationsregel $alTR_9$ überführt die Attributsfestlegung nach AnyLogic. Die Simulationsumgebung bietet jedoch kein Element an, um ausschließlichen Attributen festzulegen. Stattdessen lassen sich bei jedem Element bei den Ein- und Ausgängen Methoden hinterlegen, die eine Festlegung der Attribute der Marke erlauben.

Fast alle Elemente weisen ein besonderes Verhalten aus. Für die Attributsfestlegung kann daher nur das Service und Delay Element verwendet werden. Im ProSiT Konzept wird das Service Element ausschließlich für Aktivitäten verwendet. Eine Attributsfestlegung muss entsprechend in ein Delay Element überführt werden.

Hierfür muss zunächst die begrenzte Kapazität des Delay Elements ausgeschaltet werden, in dem das Attribute „Maximum capacity“ ausgewählt wird. Die „Delay Time“ ist wie bei der zeitabhängigen Aktivität, in der Transformationsregel $alTR_4$ mit dem Wert 0 zu versehen. Da Marken in AnyLogic mittels Java Klassen definiert werden und die Attribute entsprechende set- und get-Methoden aufweisen, wird beim Ausgang („On exit“) eine Set-Methode hinterlegt, welche das Attribute mit dem angegebenen Wert versieht. Wird bei der Festlegung eines Attributs auf ein anderes Attribut zurückgegriffen, wird eine entsprechende get-Methode aufgerufen. Die beiden in Tabelle 27 aufgeführten Beispiele orientieren sich an der automatischen Normalisierungsregel aNR_3.

Tabelle 27: Attributsfestlegung in AnyLogic

Attribute	Wert	On exit im Delay Element
durchlauf	1	entity.setDurchlauf(1);
durchlauf	durchlauf - 1	entity.setDurchlauf(entity.getDurchlauf() - 1);

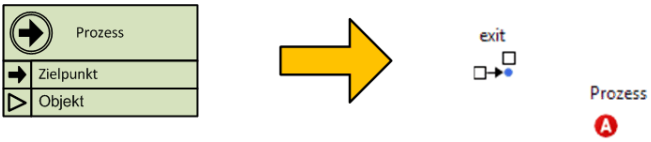
Darstellung der Transformationsregel

Transformationsregel alTR$_{10}$
If the element is a jump point, then it will be an exit element and the target process model will be added to the model. If the object of the process is different to the object of the target process, than the object is changed to the object of the target process.

Bei AnyLogic werden die einzelnen ProSiT Ablaufdiagramme in separate Modelle überführt. Der Sprungpunkt, der mit der Transformationsregel alTR$_{10}$ adressiert wird, verweist auf ein separates Modell in AnyLogic. Der Sprungpunkt selbst wird in ein Exit Element überführt und benötigt auf der anderen Seite ein Enter Element, welches durch die Transformationsregel alTR$_{11}$ erzeugt wird. Damit auf dieses Enter Element zurückgegriffen werden kann, muss das entsprechende Modell des ProSiT Ablaufdiagramms, in das Modell des Exit Elements aufgenommen werden. Damit kann im Attribute „On exit" des Exit Elements die Überführung in das Zielmodell erfolgen, in dem der Prozess und das Enter Element verwendet werden.

Die zweite Prüfung der Transformationsregel alTR$_{10}$ setzt an diesem Punkt an. Wenn der Zielprozess ein anderes Prozessobjekt verwendet, dann muss dieses entsprechend für das Simulationsmodell angepasst werden. Dieser Wechsel wird erzeugt durch die Übergabe einer neuen Marke, welche den Typ des Prozessobjekts des Zielprozesses hat. Dementsprechend ist das „On exit" Attribut mit dem Wert „[Prozess].[Zielpunkt].take(new [Prozess.Prozessobjekt]) zu versehen. Ist das Prozessobjekt hingegen identisch, wird das Attribut „On exit" mit dem Wert „Prozess.Zielpunkt.take(entity)" festgelegt.

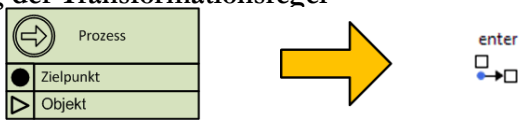
Darstellung der Transformationsregel



Transformationsregel alTR$_{11}$
If the element is a target point, then it will be an enter element.

Transformationsregel alTR$_{11}$ überführt das Gegenstück zur Transformationsregel alTR$_{10}$, den Zielpunkt, nach AnyLogic. Ein Enter Element entspricht dem Zielpunkt. Das Prozessobjekt wird nicht im Enter Element umgewandelt, sondern im Exit Element, im Rahmen der Transformationsregel alTR$_{10}$. Der Zielpunkt wird daher lediglich in das Enter Element überführt.

Darstellung der Transformationsregel



Transformationsregel alTR$_{12}$
If the element is a sub process start event, then it will be a port.

Wird ein Model in AnyLogic als ein Teilprozess verwendet, muss diesen einen Ein- und Ausgang haben. Der Eingang im Teilprozess wird durch die Transformationsregel alTR$_{12}$ erzeugt. Hierfür wird ein Teilprozessstartereignis in ein Port Element überführt. Erreicht eine Marke den Teilprozess, so wird diese im Teilprozess an den Nachfolger des Port Elements weitergeleitet.

Darstellung der Transformationsregel

Transformationsregel $alTR_{13}$
If the element is a sub process end event, then it will be a port.

Analog zur Transformationsregel $alTR_{12}$ wird ein Teilprozessendereignis durch Transformationsregel $alTR_{13}$ in ein Port Element umgewandelt. Wenn eine Marke diesen Port erreicht, dann verlässt diese den Teilprozess und wird an den Nachfolger des Teilprozesses weitergeleitet.

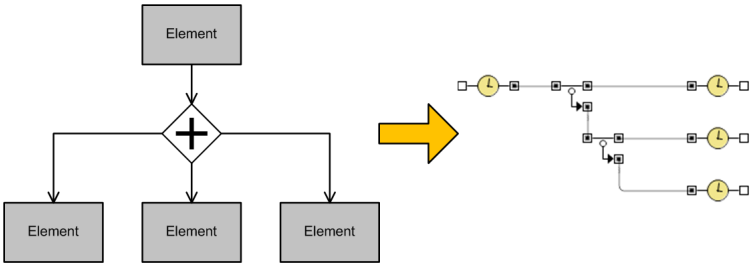
Darstellung der Transformationsregel

Transformationsregel $alTR_{14}$
If a parallel gateway has one predecessor and more than one successor then it will be a series of split elements in the quantity of the number of the successors minus one.

Die Transformationsregel $alTR_{14}$ überführt ein verzweigendes paralleles Gateway in eine Gruppe von Split Elementen. Jedes Split Element erzeugt Kopien einer Marke. Jede Kopie wird aber über den zweiten Ausgang weitergeleitet. Eine parallele Bearbeitung mit einem einzelnen Split Element ist damit nur für zwei Nachfolger realisierbar.

Wird an den zweiten Ausgang ein weiteres Split Element angefügt, dann wird über dieses eine weitere Kopie erzeugt. Durch die Verschachtlung der Split Elemente können beliebig viele Prozesspfade parallel ausgeführt werden. Für mehrere parallel auszuführende Prozesspfade sind verschachtelte Split Elemente notwendig. Damit die kopierten Marken das gleiche Prozessobjekt ausweisen, muss die Marke mit der Option „New entity copy“ erzeugt werden.

Darstellung der Transformationsregel

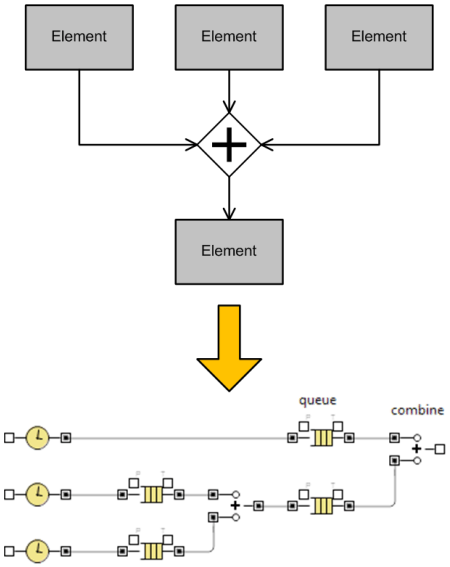


Transformationsregel $alTR_{15}$
If a parallel gateway has more than one predecessor and one successor, then it will be a series of combine elements in the number of the predecessors minus one and a queue is inserted before every entry of the combine elements.

Die Transformationsregel $alTR_{15}$ überführt ein zusammenführendes paralleles Gateway nach AnyLogic. Um Prozesspfade zusammenzuführen ist das Combine Element notwendig. Dieses kann zwei Vorgänger miteinander verschmelzen. Ähnlich zur Transformationsregel $alTR_{14}$ müssen die Combine Elemente verschachtelt werden.

An jedem Eingang des Combine Elements kann aber nur eine Marke zu zeitgleich sein. Andersfalls führt dies bei einem Simulationslauf zu einem Laufzeitfehler. Um dies zu verhindern, muss vor jedem Eingang eines Combine Elements ein Queue Element eingefügt werden. Eine Synchronisation von Marken mit gleicher Seriennummer kann somit nicht erfolgen. Für einen problemlosen Simulationsablauf müssen die Queue Elemente ihr Attribute „Maximum capacity“ aktivieren. Bei den Combine Elementen erhält das Attribut „The resulting entity is“ den Wert „entity1“. Dies ist notwendig, damit die zusammengeführte Marke die zuvor festgelegten Attributswerte aufweist.

Darstellung der Transformationsregel



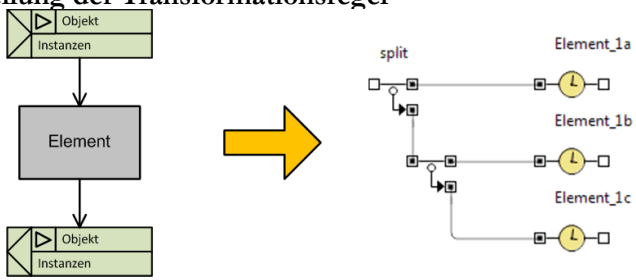
Transformationsregel alTR$_{16}$
If the element is a splitting instance and it has not a corresponding merging instance, then it will be a split element. If the element is a splitting instance and it has a corresponding merging instance, then it will be a series split elements in the number of the instances and elements between these two instance elements will be duplicated for each process path.

Die Transformationsregel alTR$_{16}$ ist zweigeteilt. Eine verzweigende Instanziierung kann prinzipiell mittels eines Split Elements umgesetzt werden. Dieses erzeugt am zweiten Ausgang die gewünschte Anzahl an Kopien. Werden beide Ausgänge mit dem gleichen Nachfolger verbunden, dann ist das Verhalten einer verzweigenden Instanziierung abgebildet. Die neu erzeugten Kopien („New entity copy“) müssen dem Prozessobjekt entsprechen, um Fehler im Simulationslauf zu vermeiden. Mit dem Attribut „Number of copies“ werden die Anzahl der Instanzen $i-1$ eingetragen.

Darstellung der Transformationsregel

Ein Problem ergibt sich in Kombination mit dem Gegenstück, der zusammenführenden Instanziierung. Diese kann wie in der nachfolgend dargestellten Transformationsregel alTR$_{17}$ eine Synchronisation nur über mehrere eingehende Prozesspfade umsetzen. Hat eine verzweigende Instanziierung keine korrespondierende zusammenführende Instanziierung kann der beschriebene Fall verwendet werden. Bei einer enstprechenden zusammenführenden Instanziierung müssen mehrere Split Elemente verschachtelt werden, ähnlich der Transformationsregel alTR$_{14}$. Als Konsequenz aus dieser Umsetzung muss jedes Element zwischen dem Instanziierungspaar für jeden Prozesspfad dupliziert werden. Exemplarisch ist dies in der nachfolgenden Darstellung durch die Nummerierung mit dem Kleinbuchstaben angedeutet.

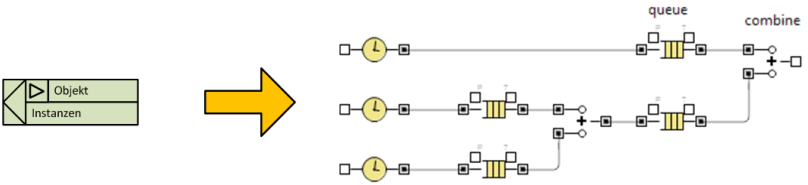
Darstellung der Transformationsregel



Transformationsregel alTR$_{17}$
If the element is a merging instance and it has a corresponding splitting instance element, then it will be a series of combine elements in the number of the predecessors minus one and a queue is inserted between the predecessors and the entries of the combine elements. Each parallel path, that is created by a corresponding splitting instance is connected to an individual queue element.

Transformationsregel alTR$_{17}$ führt mehrere parallel ausgeführte Prozesspfade zusammen, die durch eine verzweigende Instanziierung erzeugt wurden. Als Prämisse muss hierfür eine korrespondierende verzweigende Instanziierung vorhanden sein. In diesem Fall entspricht das erzeugte Konstrukt dem Ergebnis der Transformationsregel alTR$_{15}$. Jeder durch die Transformationsregel alTR$_{16}$ erzeugte parallele Prozesspfad muss mit dem Eingang eines Queue Elements verbunden werden.

Darstellung der Transformationsregel



Transformationsregel alTR$_{18}$
If an exclusive gateway has one predecessor and more than one successor, then it will be a series of select output elements and the probabilities of the exclusive keys that are the successors of the gateway will be added to the select output elements.

Transformationsregel alTR$_{18}$ überführt ein verzweigendes exklusives Gateway nach AnyLogic. Exklusive Entscheidungen werden in AnyLogic durch das Select Output Element umgesetzt. Dieses gibt es in den zwei Ausführungen, mit zwei Nachfolgern und mit fünf Nachfolgern. Eine abweichende Zahl an Nachfolgern kann jedoch durch das Schachteln des Select Output Elements realisiert werden. Um eine klare Transformationsregel zu definieren, wird ausschließlich das Select Output Element verwendet, welches zwei Nachfolger anbietet. Bei n Nachfolgern werden $n-1$ Select Output Elemente benötigt.

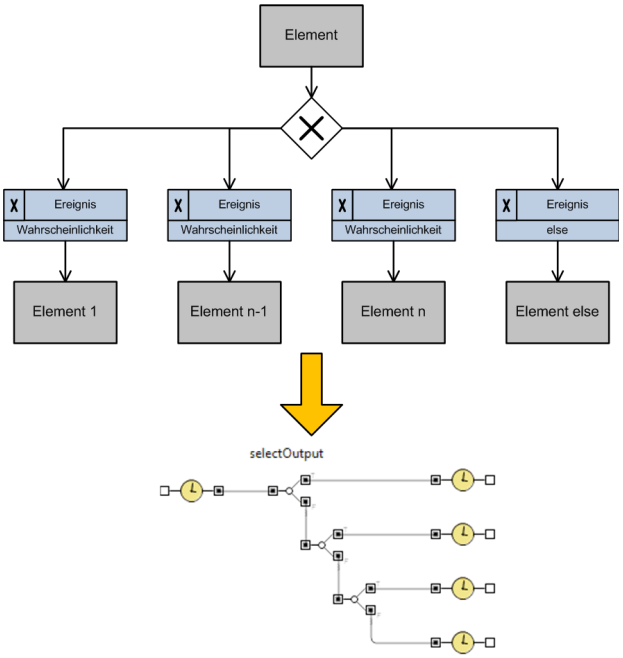
Die Wahrscheinlichkeit für die Ausführung eines Pfades kann nicht direkt aus den exklusiven Schlüsseln übernommen werden. Wird ein else-Schlüssel verwendet, dann erhält dieser die Wahrscheinlichkeit, die sich aus der Subtraktion von 1 und der Summe der angegebenen Wahrscheinlichkeiten ergibt. Zur Verdeutlichung der Berechnung wird ein Beispiel mit vier Nachfolgern angenommen. Diese weisen zwei Mal 20 sowie zwei Mal 30 Prozent Wahrscheinlichkeit für die Durchführung auf. Durch die 4 Nachfolger werden entsprechend 3 Select Output Elemente benötigt, deren Wahrscheinlichkeiten (Attribut „probability“) in Tabelle 28 aufgeführt sind.

Tabelle 28: Wahrscheinlichkeiten bei drei Select Output Elementen in AnyLogic

Element	Wahrscheinlichkeit (true)	Wahrscheinlichkeit (false)
Select Output 1	0,2	0,8
Select Output 2	0,25	0,75
Select Output 3	0,5	0,5

Beim ersten Select Output Element wird in 20 von 100 Fällen der erste Nachfolger ausgeführt. 80% der Fälle werden an den zweiten Nachfolger weitergeleitet. In 20% der 80%-Fälle wird der zweite Nachfolger ausgeführt. Um die Wahrscheinlichkeit im zweiten Select Output Element anzugeben, ist eine Normalisierung auf 100% notwendig. Hierfür werden die 0,2 mit den 0,8 dividiert, woraus sich die 0,25 des zweiten Select Output Elements ergeben. Die Wahrscheinlichkeit des ersten Pfades im Select Output Element n ist somit abhängig von der Wahrscheinlichkeit des nicht Eintretens des $n-1$ Select Output Elements.

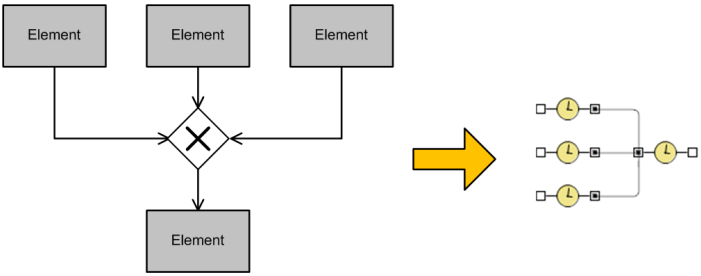
Darstellung der Transformationsregel



Transformationsregel $alTR_{19}$
If an exclusive gateway has more than one predecessor and one successor, then it will be removed and the predecessors of the gateway will get the successor of the gateway as their successor.

Die Transformationsregel $alTR_{19}$ überführt die Logik das zusammenführende exklusive Gateway nach AnyLogic. Ein zusammenführendes Element ist aber in AnyLogic nicht vorhanden. Die Vorgänger des Gateways erhalten daher den Nachfolger des Gateways als Nachfolger.

Darstellung der Transformationsregel

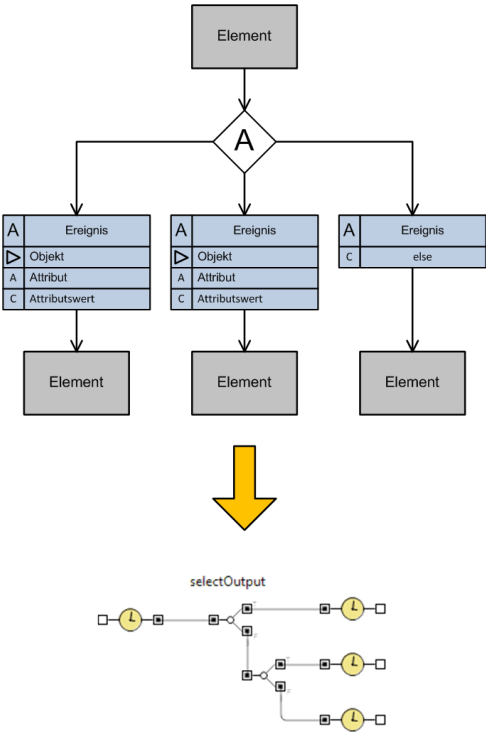


Transformationsregel $alTR_{20}$
If an attribute-based gateway has one predecessor and more than one successor, then it will be a series of select output elements and the attributes of the attribute-based keys that are the successors of the gateway will be added to the select output elements.

Transformationsregel $alTR_{20}$ überführt ein verzweigendes attributsbasiertes Gateway nach AnyLogic. Wie beim verzweigenden exklusiven Gateway, kann dieses nicht mit einem einzelnen Element realisiert werden. Analog zur Transformationsregel $alTR_{18}$ werden Select Output Elemente verschachtelt, womit $n-1$ Elemente bei n Nachfolgern benötigt werden. Für jedes Select Output Element wird das Attribut „Select True output“ mit dem Wert „If condition is true“ versehen. Damit kann in das Attribut „Condition“ die Bedingung aus dem attributsbasierten Schlüssel eingetragen werden. Die Form ist hierbei „entity.get[Attribut]() == [Wert]“.

Durch die zwingende Anwendung der Normalisierungsregel sNR_{11} hat ein attributsbasiertes Gateway immer einen else-Schlüssel. Dieser wird im am tiefsten verschachtelten Select Output Element eingebunden, in dem es an den else-Zweig angebunden wird. Hierdurch wird dieser Pfad nur ausgeführt, wenn keine vorherige Prüfung wahr ist.

Darstellung der Transformationsregel



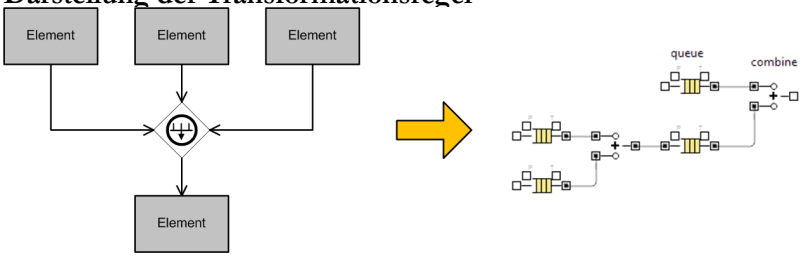
Transformationsregel alTR$_{21}$
If an attribute-based gateway has more than one predecessor and one successor, then it will be removed and the predecessors of the gateway will get the successor as their successor.

Transformationsregel alTR$_{21}$ arbeitet analog zur Transformationsregel arTR$_{19}$. Das attributsbasierte Gateway wird entfernt und die Vorgänger werden mit dem Nachfolger verknüpft.

Transformationsregel $alTR_{22}$
If an inclusive gateway or a wait-for-all inclusive gateway has more than one predecessor and one successor, then it will be a series of combine elements in the number of the predecessors minus one and a queue is inserted between the predecessors and the combine elements.

Die Transformationsregel $alTR_{22}$ überführt ein nicht klassifiziertes oder inklusives wait-for-all Gateway nach AnyLogic. Semantisch erfolgt eine Synchronisation der Marken für alle – durch ein inklusives Gateway – ausgeführten Prozesspfade. Um dieses Verhalten abzubilden, wird ein Modellkonstrukt erzeugt, das mit dem des zusammenführenden parallelen Gateways identisch ist. Für die Erläuterung der Regel wird auf die Transformationsregel $alTR_{15}$ verwiesen.

Darstellung der Transformationsregel



Transformationsregel $alTR_{23}$
If a first-come inclusive gateway has more than one predecessor and one successor, then it will be a select output element. The first successor will be a sink element and the second successor will be the successor of the inclusive gateway.

Transformationsregel $alTR_{23}$ überführt das zusammenführende inklusive first-come Gateway in ein Select Output Element. Bei diesem ist der erste Nachfolger ein Sink Element und der zweite Nachfolger der Nachfolger des Gateways. Mit dem Select Output Element werden keine zwei Marken mit gleicher Id an den Nachfolger weitergereicht. Um diese Prüfung sicher zu stellen, werden zwei Klassen definiert „EverytimeMap“ und „EverytimeTable“. Die Klassen, die für ein

bestimmtes Verhalten in AnyLogic definiert sind, werden im Anhang ab Seite 488 aufgeführt.

Die Klasse EvertimeTable verwaltet für ein zusammenführendes Gateway alle Identifikationsnummern, die dieses bereits passierten. Die entscheidende Methode für die Realisierung dieses Verhaltens ist in der Klasse „EverytimeMap“ definiert. Der Aufruf der Methode erfolgt über das Split Element im Attribut Condition mit dem Aufruf EvertimeMap.getInstance().checkEntityId("A", entity.getId()). Für die Ermöglichung dieses Aufrufs muss zusätzlich im Attribut „Entity class“ der Wert „ProSiT Entry“ hinterlegt werden.

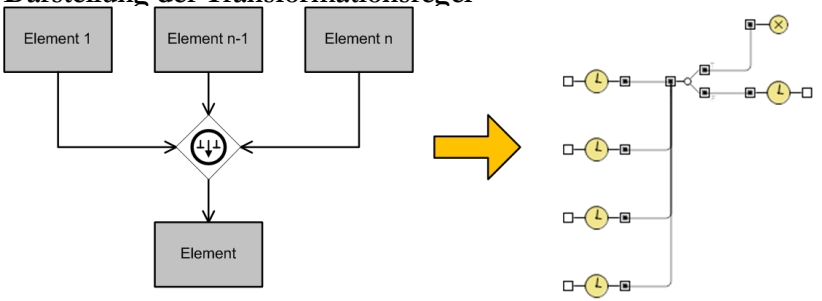
Methode „checkEntryId“ der Klasse „EverytimeMap“

```
public boolean checkEntityId(String decision, long entryId) {
  EverytimeTable table = everytimeTables.get(decision);
  if ( table == null ) {
    table = new EverytimeTable();
    table.addId(entryId);
    everytimeTables.put(decision, table);

    return false;
  }

  if ( table.hasId(entryId) == false ) {
    table.addId(entryId);
    return false;
  } else
    return true;
}
```

Die Methode „checkEntityId“ gibt den Wert true zurück, wenn eine Marke mit der gleichen Identifikationsnummer das Gateway bereits passierte, womit die Marke durch das Sink Element aus dem Simulationsmodell entfernt wird. Als Erstes wird in der Methode geprüft, ob es für die Zusammenführung bereits eine Tabelle gibt. Ist dies nicht der Fall, dann wird eine neue Tabelle erzeugt, die Identifikationsnummer in dieser aufgenommen und die Tabelle dem Singleton „EverytimeMap“ hinzugefügt. Entsprechend dieses Ablaufs gibt die Methode den Wert false zurück. Ist für die Zusammenführung bereits eine Tabelle hinterlegt, wird diese durch die zweite If-Abfrage nach dem Vorhandensein der Identifikationsnummer überprüft. Ist

diese nicht enthalten, dann wird die Nummer hinzugefügt und ebenfalls der Wert false zurückgegeben. Wenn aber die Identifikationsnummer in der Tabelle vorhanden ist, wird über den else-Zweig true zurückgegeben, womit als nächster Schritt im Simulationsmodell die Marke aus dem Modell entfernt wird.

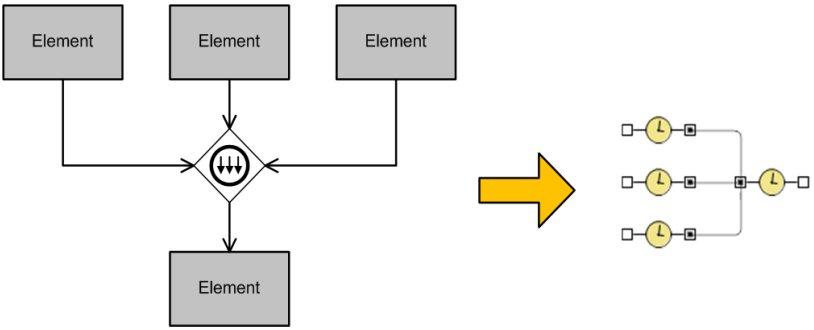
Darstellung der Transformationsregel



Transformationsregel alTR$_{24}$
If an every-time inclusive gateway has more than one predecessor and one successor, then it will be removed and the predecessors of the gateway will get the successor as their successor.

Die Transformationsregel alTR$_{24}$ überführt das zusammenführende inklusive every-time Gateway in die Simulationsumgebung AnyLogic. Im Gegensatz zu den anderen zwei Transformationsregeln für zusammenführende inklusive Gateways alTR$_{22}$ und alTR$_{23}$ erfolgt keine Verarbeitung, wenn die Marken das Gateway erreichen. Jede ankommende Marke wird an den Nachfolger des Gateways weitergereicht. Hierfür wird das Gateway entfernt und die Vorgänger werden mit dessen Nachfolger verknüpft. Das erzeugte Konstrukt in der Simulationsumgebung entspricht dem der Transformationsregeln alTR$_{19}$ und alTR$_{21}$.

Darstellung der Transformationsregel



Transformationsregel alTR$_{25}$
If an inclusive gateway or a wait-for-all inclusive gateway has one predecessor and more than one successor, then it will be a construct of several elements. Let n be the number the number of paths that follow the Gateway. Every path until $n-1$ gets a split element that has as its first successor a select output element. The second exit of every select output element until the second to last is connected to the next select output element and the last is connected to the select output element n. The select output elements have as their first successor the corresponding successor of the inclusive key and as their second successor the corresponding queue of the wait-for-all inclusive gateway for the corresponding path.

Transformationsregel alTR$_{25}$ überführt das verzweigende inklusive Gateway oder das verzweigende klassifizierte inklusive wait-for-all Gateway nach AnyLogic. Da die einzelnen Pfade des inklusiven Gateways, wie bei einem parallelen Gateway ausgeführt werden, ist das Duplizieren der Marke notwendig. Wie bei der Transformationsregel alTR$_{14}$ sind bei n Nachfolgern $n-1$ Split Elemente notwendig. Bis zum vorletzten Split Element ist der zweite Ausgang jeweils mit dem nächsten Split Element verbunden. Der erste Ausgang jedes Split Element ist mit einem Select Output Element verknüpft und das letzte Split Element noch zusätzlich mit dem zweiten Ausgang. Damit werden n Select Output Elemente erzeugt, womit jeder inklusive Schlüssel durch ein Select Output Element repräsentiert wird. An den ersten Ausgang, wenn die Prüfbedingung wahr ist, folgt der Nachfolger des inklusiven Schlüssels. Da ein verzweigendes inklusives wait-for-all

Gateway nur im Zusammenhang mit einem entsprechenden zusammenführenden inklusiven Gateway verwendet werden kann, hat jedes Select Output Element das entsprechende Queue Element als Nachfolger. Gemäß Transformationsregel alTR$_{22}$ sind hierfür $n-1$ Combine Elemente notwendig.

Neben diesen Elementen müssen, für die Umsetzung des korrekten Ablaufs, Klassen und Methoden in AnyLogic definiert werden. Die notwendigen Klassen werden im Anhang ab Seite 488 aufgeführt. Zu Beginn des Simulationsvorhabens muss die Wahrscheinlichkeitstabelle aufgebaut werden. Hierfür werden zwei Klassen definiert eine „DecisionMap“, welche die einzelnen Entscheidungstabellen enthält und die eigentlichen Entscheidungstabellen mit der Klasse „DecisionTable“. Die wichtigste Methode „getPath“ der Klasse „DecisionMap“ ist nachfolgend aufgeführt:

Methode „getPath“ der Klasse „DecisonMap“

```
public String getPath(String id) {
  if ( this.decisionTables.containsKey(id) == false ) {
    return null;
  }

  DecisionTable table = decisionTables.get(id);

  float probability = new Random().nextFloat() % 1;
  return table.getPaths(probability);
}
```

Diese Methode liefert mittels eines Delegate Entwurfsmusters die auszuführenden Prozesspfade einer gewünschten Entscheidungstabelle zurück. Diese Methode bedarf einer Weiterentwicklung, da der Zufallsgenerator von Java verwendet wird. Bei diesem kann nicht sichergestellt werden, ob bei einem Simulationslauf mit den gleichen Einstellungen, das gleiche Ergebnis erzeugt wird. Dementsprechend ist für diese Methode beziehungsweise für eine Entscheidungstabelle ein simulationssicherer Zufallsgenerator zu verwenden oder zu konzipieren.

Methode „getPaths“ der Klasse „DecisionTable“

```
public String getPaths(float probability) {
  if ( probability < 0.0f && probability > 1.0f )
    return null;

    int index = 0;

    for ( float p = 0.0f; p<probability; index++ ) {
      p += this.cases.get(index).getProbability();
    }

    return this.cases.get(index-1).getPath();
}
```

Die Klasse „DecisionTable“ stellt eine erweiterbare Tabelle dar, dessen Zeilen in der Klasse „Case“ definiert sind. Somit werden die Spalten der Tabelle in der Klasse „Case“ verarbeitet und die Anzahl der Zeilen in der Klasse „DecisionTable“. Mittels der Methode „valid“ kann die Gültigkeit der Entscheidungstabelle überprüft werden, in dem ermittelt wird, ob die Summe aller Wahrscheinlichkeiten 1 beträgt. Die Methode „getPaths“ nimmt eine Wahrscheinlichkeit entgegen, die den Wert zwischen 0 und 1 aufweist. Durch eine Addition der Wahrscheinlichkeiten für die einzelnen Pfade wird das Pfadbündel ausgewählt, das sich summiert in der angegebenen Wahrscheinlichkeit befindet. Der Ablauf dieser Methode orientiert sich an der Wahrscheinlichkeitsstabelle wie diese in AnyLogic umgesetzt wird. Die Klasse „Case“ beinhaltet lediglich zwei Attribute: Für den auszuführenden Pfad und dessen Wahrscheinlichkeit.

Zur Initialisierung der Entscheidungstabellen ist eine Methode im Attribute „Anlaufcode“ des Simulationsmodells zu hinterlegen. Diese muss die Funktion für die Initialisierung der Entscheidungstabelle aufrufen. Exemplarisch ist dies mit der nachfolgenden Methode „initDecisionMap“ aufgeführt. Die im Quellcode abgebildete Wahrscheinlichkeitstabelle entspricht der Tabelle 14 auf Seite 79.

Methode „initDecisionMap“

```
public static void init() {
  DecisionTable table = new DecisionTable();
  table.addCase(new Case("1",   0.10f));
  table.addCase(new Case("2",   0.05f));
  table.addCase(new Case("3",   0.15f));
  table.addCase(new Case("12",  0.25f));
  table.addCase(new Case("23",  0.30f));
  table.addCase(new Case("123", 0.05f));
  table.addCase(new Case("0",   0.10f));

  DecisionMap map = DecisionMap.getInstance();
  map.addDecisionTable("A", table);
}
```

Um die Entscheidungstabelle im Rahmen der Transformationsregel $alTR_{25}$ zu verwenden, wird die Klasse „ProSiTEntity“ benötigt. Von dieser Klasse werden alle Objekte des Transformationsmodells abgeleitet.

Methode „setOrDecision“ der Klasse „ProSiTEntity“

```
public void setOrDecision(String decision, String paths) {
  pathTable.put(decision, paths);
}
```

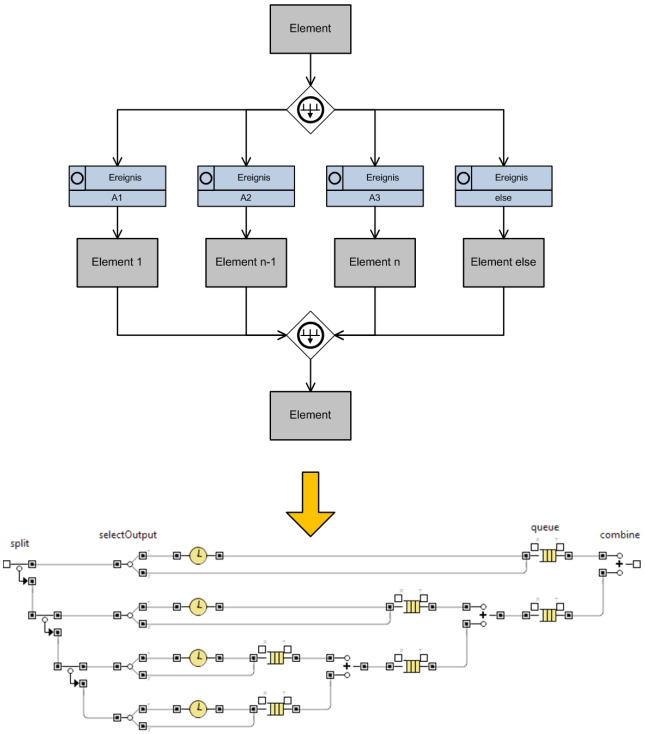
Die Methode „setOrDecision“ wird beim ersten Split Element im Attribute „On enter“ mit der beispielhaften Ausprägung „entity.setOrDecision("A", DecisionMap.getInstance().getPath("A"));“ verwendet. Neben diesem Attribut im ersten Split Element, wird bei jedem Split Element der Wert „Entity“ des Attribute „Entity classes“ mit dem Wert „ProSiTEntity“ ersetzt. Entsprechend wird bei der Kopie („new entity (copy)“) eine neue Instanz der „ProSiTEntity“ erzeugt („new ProSiTEntity()“). Damit diese jedoch die Werte der Entscheidungstabellen erhält wird im Attribut „On exit copy“ der Wert „entity.copy(original)“ verwendet. Die Entscheidung, ob ein Prozesspfad ausgeführt wird, ist in den selectOutput Elementen hinterlegt. Durch die Angabe „ProSiTEntity“ im Attribut „Entity class“ kann auf die Methoden der Klasse zurückgegriffen werden. Als „Condition“ kann damit „entity.hasPath(„A“, '1') angegeben werden. Bei dieser Methode wird die Zeichenkette mit den durchzuführenden

Prozesspfaden sequenziell durchlaufen und nach dem angegebenen Prozesspfad geprüft. Da der Rückgabewert der Methode ein Wahrheitswert ist, muss das Attribut „Select True output“ auf „If condition is true“ eingestellt werden.

Methode „hasPath“ der Klasse „ProSiTEntity“

```
public boolean hasPath(String decision, char path) {
  if ( pathTable.containsKey(decision) == false ) {
    return false;
  }

  String paths = pathTable.get(decision);

  for ( int i=0; i<paths.length(); i++ ) {
    char nextPath = paths.charAt(i);
    if ( nextPath == path ) {
      return true;
    }
  }

  return false;
}
```

Darstellung der Transformationsregel



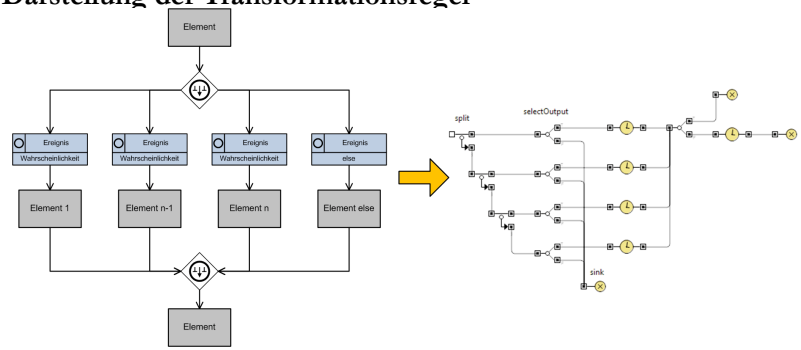
Transformationsregel $alTR_{26}$
If a first-come inclusive gateway has one predecessor and more than one successor, then it will be a construct of several elements. Let n be the number the number of paths that follow the Gateway. Every path until $n-1$ gets a split element that has as its first successor a select output element. The second exit of every select output element until the second to last is connected to the next select output element and the last is connected to the select output element n. The select output elements have as their first successor the corresponding successor of the inclusive key and as their second successor a sink element.

Die Transformationsregel $alTR_{26}$ ähnelt der Transformation aus der Regel $alTR_{25}$. Der Unterschied liegt jedoch in den zweiten Ausgängen bei den Select Output Elementen. Diese sind mit einem Sink Element

verknüpft, um die Marke aus dem Simulationsmodell zu entfernen, wenn der Prozesspfad nicht ausgeführt wird. Im Sinne eines inklusiven Oders muss jeder Pfad geprüft werden, ob dieser auszuführen ist. Hierfür werden ebenfalls bezogen, auf die Transformationsregel alTR$_{14}$, bei n inklusiven Schlüsseln $n-1$ Split Elemente erzeugt. Alle Split Elemente, mit Ausnahme des Letzten sind jeweils mit dem nachfolgenden Split Element über den zweiten Ausgang verknüpft. Das letzte Split Element ist mit einem weiteren Split Element verknüpft, sodass n Split Elemente, für jeden Prozesspfad eines, vorliegen. Die Split Elemente prüfen jeweils, ob der Prozesspfad auszuführen ist. Daher ist der erste Ausgang mit dem Nachfolger des jeweiligen inklusiven Schlüssels verknüpft. Da jedoch nur die ausgeführten Prozesspfade weiterverarbeitet werden sollen, ist an den zweiten Ausgang jedes Split Elements ein Sink Elemente angefügt, welches die Marken für nicht ausgeführte Prozesspfade entfernt.

Die festzulegenden Attribute und dafür notwendigen Klassen entsprechen denen der Transformationsregel alTR$_{26}$. Die Erläuterung ist entsprechend bei dieser Transformationsregel vorzufinden.

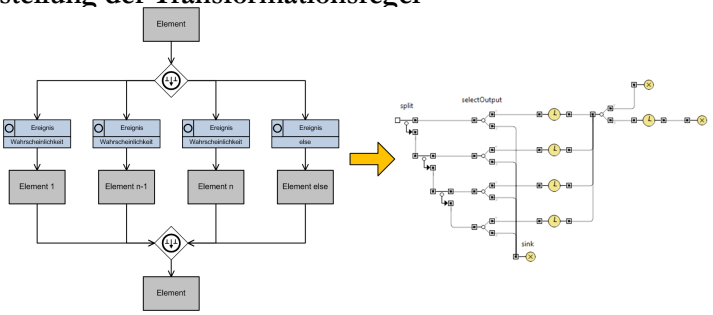
Darstellung der Transformationsregel



Transformationsregel alTR$_{27}$
If an every-time inclusive gateway has one predecessor and more than one successor, then it will be a construct of several elements. Let n be the number the number of paths that follow the Gateway. Every path until $n-1$ gets a split element that has as its first successor a select output element. The second exit of every select output element until the second to last is connected to the next select output element and the last is connected to the select output element n. The select output elements have as their first successor the corresponding successor of the inclusive key and as their second successor a sink element.

Transformationsregel alTR$_{27}$ erzeugt bei der Überführung eines inklusives every-time Gateways nach AnyLogic mehrere Elemente. Das erzeugte Konstrukt entspricht weitgehend dem Ergebnis der Transformationsregel alTR$_{24}$.

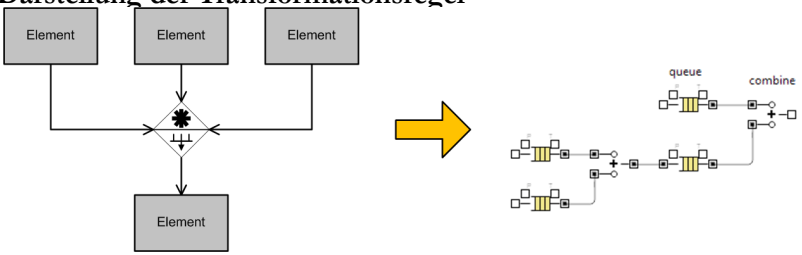
Darstellung der Transformationsregel



Transformationsregel alTR$_{28}$
If a complex gateway or a wait-for-all complex gateway has more than one predecessor and one successor, then it will be a series of combine elements in the number of the predecessors minus one and a queue is inserted between the predecessors and the combine elements.

Die Transformationsregel alTR$_{28}$ entspricht semantisch der Transformationsregel alTR$_{22}$. Beide führen eine Synchronisation für inklusive erzeugte Prozesspfade durch. Eine Unterscheidung zwischen wahrscheinlichkeitsbedingter und konditionsbasierter Verzweigung ist für die Synchronisation nicht von Bedeutung. Daher wird für die Erläuterung der Transformationsregel auf die Transformationsregel alTR$_{22}$ verwiesen.

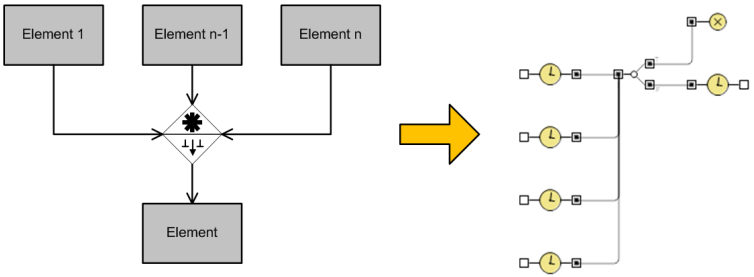
Darstellung der Transformationsregel



Transformationsregel alTR$_{29}$
If a first-come complex gateway has more than one predecessor and one successor, then it will be a select output element. The first successor will be a sink element and the second successor will be the successor of the complex gateway.

Das zusammenführende komplexe first-come Gateway unterscheidet sich nicht vom zusammenführenden inklusiven first-come Gateway. Für die Erläuterung der Transformationsregel alTR$_{29}$ wird daher auf die Transformationsregel alTR$_{23}$ verwiesen.

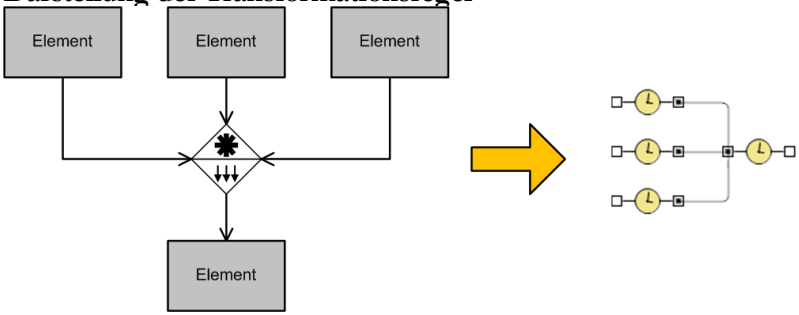
Darstellung der Transformationsregel



Transformationsregel $alTR_{30}$
If an every-time complex gateway has more than one predecessor and one successor, then it will be removed and the predecessors of the gateway will get the successor as their successor.

Wie bei Transformationsregel $alTR_{24}$ erfolgt bei einem komplexen every-time Gateway keine Verarbeitung der ankommenden Marken. Stattdessen wird jede ankommende Marke an den Nachfolger des Gateways weitergereicht. Durch die Transformationsregel $alTR_{30}$ wird das Gateway entfernt und die Vorgänger erhalten als Nachfolger den Nachfolger des Gateways.

Darstellung der Transformationsregel



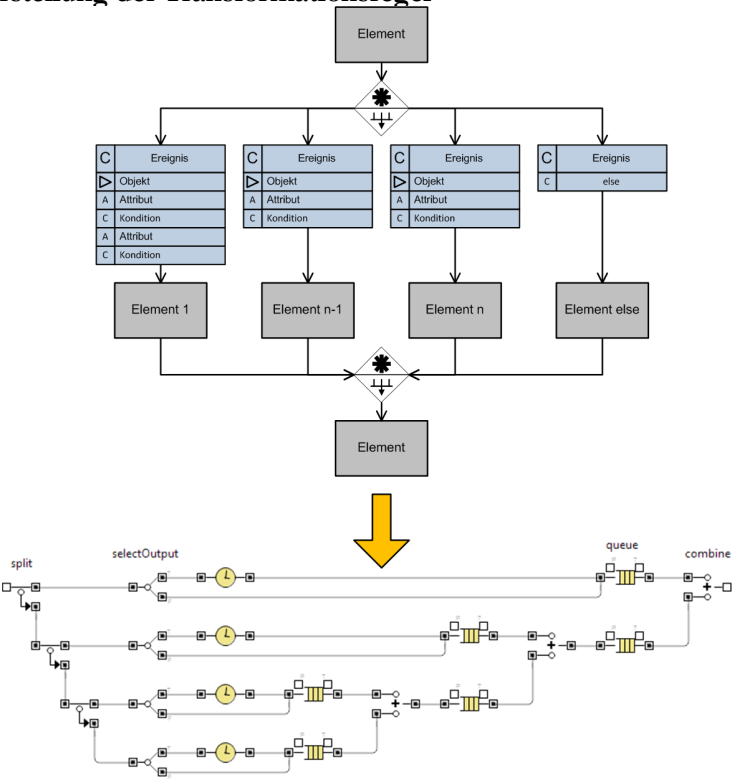
Transformationsregel alTR$_{31}$
If a complex gateway or a wait-for-all complex gateway has one predecessor and more than one successor, then it will be a construct of several elements. Let n be the number of paths that follow the Gateway. Every path until $n-1$ gets a split element that has as its first successor a select output element. The second exit of every select output element until the second to last is connected to the next select output element and the last is connected to the select output element n. The select output elements have as their first successor the corresponding successor of the inclusive key and as their second successor the corresponding queue of the wait-for-all complex gateway for the corresponding path.

Transformationsregel alTR$_{31}$ erzeugt für ein verzweigendes komplexes oder ein verzweigendes komplexes wait-for-all Gateway das gleiche Konstrukt, auf Element Ebene, wie Transformationsregel alTR$_{25}$. Aufgrund dieser Gleichheit wird für die Erläuterung der Elemente auf die Transformationsregel alTR$_{25}$ verwiesen und nur auf die Attribute eingegangen.

Alle Split Elemente weisen die gleichen drei Attributswerte auf. Das Attribut „Entity classes“ wird mit dem „ProSiTEntity“ Eintrag versehen, damit auf die Methoden dieser Klasse zurückgegriffen werden kann. Entsprechend erfolgt die Erzeugung einer neuen ProSiTEntity („new ProSiTEntity()“) im Attribut „New entity (copy)“. Ebenfalls erfolgt eine Kopie des Originals über die Methode „entity.copy(original)“, die den Ausgang der Kopie („On exit copy“) verlässt. Die Select Output Elemente werden beim Attribut „Select True output“ mit dem Wert „IF condition is true“ hinterlegt und die „Entity class“ wird auf das „ProSiTEntity“ Element umgestellt. Die Konditionen, die im komplexen Schlüssel definiert sind, werden nach dem Muster „entity.get[Attribut]() [C] [Kondition]“ hinterlegt. Ist mehr als eine Kondition im komplexen Schlüssel definiert, dann werden diese mit einem logischen Und aus Java „&&“ verbunden. Die Attribute, die in den komplexen Schlüsseln definiert sind, müssen hierfür aus der Objektsicht in die entsprechende Klasse in AnyLogic überführt werden. Die Klasse erbt die Eigenschaften der Klassen „ProSiTEntity“.

Eine besondere Überführung ist für einen komplexen else-Schlüssel notwendig. Dieser ist durch die Normalisierungsregel sNR_{12} immer nach einem verzweigenden komplexen Gateway vorzufinden. Der entsprechende Prozesspfad ist nur auszuführen, wenn kein anderer Prozesspfad, keine andere Bedingung eines komplexen Schlüssels wahr ist. Dies kann sichergestellt werden, in dem beim komplexen else-Schlüssel die mit einem logischen Und verbundenen Negationen der anderen Schlüssel geprüft werden. Aus der Prüfung A beim ersten Prozesspfad und B beim Zeiten Prozesspfad, wird so die logische Prüfung $\neg A \wedge \neg B$ im komplexen else-Schlüssel.

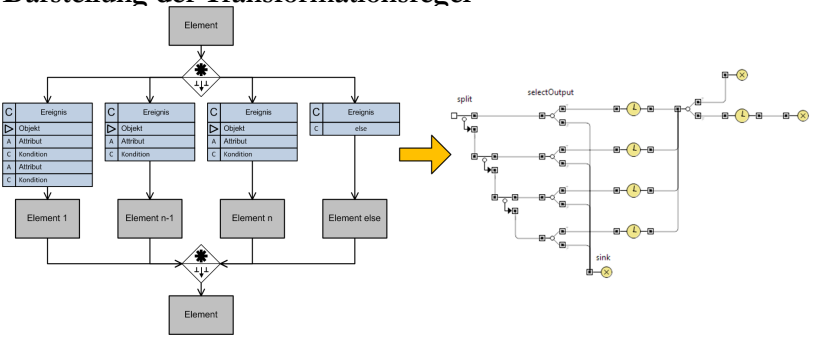
Darstellung der Transformationsregel



Transformationsregel alTR$_{32}$
If a first-come complex gateway has one predecessor and more than one successor, then it will be a construct of several elements. Let n be the number the number of paths that follow the Gateway. Every path until $n-1$ gets a split element that has as its first successor a select output element. The second exit of every select output element until the second to last is connected to the next select output element and the last is connected to the select output element n. The select output elements have as their first successor the corresponding successor of the complex key and as their second successor a sink element.

Die Transformationsregel alTR$_{32}$ überführt ein komplexes first-come Gateway nach AnyLogic. Das erzeugte Modellkonstrukt wurde bereits bei zwei anderen Transformationsregeln erläutert. Auf Elementebene entspricht das Ergebnis der Transformationsregel alTR$_{32}$ der Transformationsregel alTR$_{26}$, auf Attributsebene hingegen sind die entsprechenden Einstellungen in der vorhergehenden Transformationsregel alTR$_{31}$ aufgeführt worden. Dementsprechend wird für weitere Ausführungen bezogen auf die Transformationsregel alTR$_{32}$ auf die anderen beiden Regeln verwiesen.

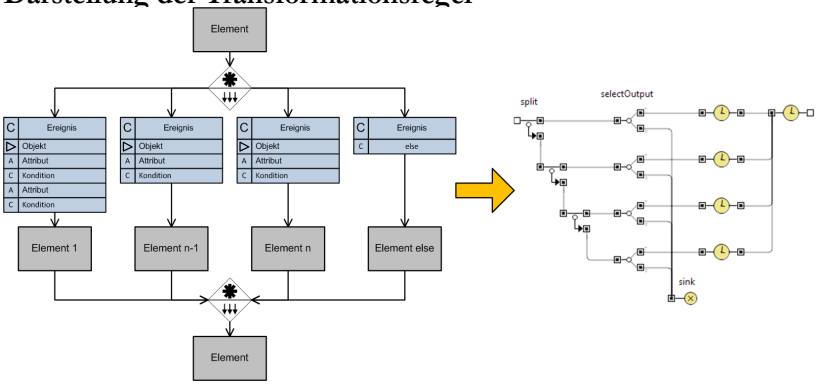
Darstellung der Transformationsregel



Transformationsregel alTR$_{33}$
If an every-time complex gateway has one predecessor and more than one successor, then it will be a construct of several elements. Let n be the number the number of paths that follow the Gateway. Every path until $n-1$ gets a split element that has as its first successor a select output element. The second exit of every select output element until the second to last is connected to the next select output element and the last is connected to the select output element n. The select output elements have as their first successor the corresponding successor of the complex key and as their second successor a sink element.

Wie bei der vorangehenden Transformationsregel wird bei der Transformationsregel alTR$_{33}$ auf zwei andere Regeln verwiesen. Da sich das Ergebnis der Transformation nicht von der Transformationsregel alTR$_{32}$ unterscheidet wird ebenfalls auf die Transformationsregel alTR$_{26}$ für die Elementebene und auf die Transformationsregel alTR$_{31}$ für die Attributsebene verwiesen.

Darstellung der Transformationsregel



Transformationsregel $alTR_{34}$
Every resource will be added as an individual resource pool element.

Die Transformationsregel $alTR_{34}$ überführt jede Ressource der Ressourcensicht nach AnyLogic. Diese werden als individuelles „Resource Pool“ Element umgesetzt. Der Name der Ressource wird als Name des Elements verwendet. Über die Namensgleichheit mit den Ressourcen, die in den Aktivitäten verwendet werden, ist eine korrekte Umsetzung der Transformationsregel $alTR_3$ sichergestellt. Die zur Verfügung stehende Quantität wird in das Attribut „Capacity“ eingetragen. Wird die zur Verfügung stehende Quantität hingegen mit einem Zeitplan spezifiziert, wie dieser beispielhaft in Tabelle 25 dargestellt wird, dann wird das Attribut „Capacity defined“ auf „by Schedule“ eingestellt und in das Feld „Capacity schedule“ der entsprechende Zeitplan eingetragen.

Transformationsregel $alTR_{35}$
Every schedule will be added as an individual schedule element.

Analog zur Transformationsregel $alTR_{34}$ überführt Transformationsregel $alTR_{35}$ jeden Zeitplan, der in der Ressourcensicht definiert ist, in ein separates Schedule Element. Der Zeitplan wird beim Attribut „Value type“ auf „integer“ und bei „Duration type“ auf „Days/Weeks“ eingestellt. Die Einträge des Zeitplans können daraufhin mit Start- und Endzeitpunkt sowie der zur Verfügung stehenden Menge überführt werden. Aus den angegebenen Zeiten muss darüber hinaus ausgelesen werden in welchen Zeiträumen sich der Zeitplan wiederholt und dies entsprechend im Attribut „Repeat every“ angeben.

Transformationsregel $alTR_{36}$
Every object will be added as a class that is the child of the ProSiTEntity class.

Die Transformationsregel $alTR_{36}$ überführt die in der Objektsicht definierten Objekte nach AnyLogic. Objekte definieren die Marken, die das Simulationsmodell durchlaufen. Diese Definition wird in AnyLogic mittels Klassen realisiert, welche von der Klasse Entity abgeleitet werden, beziehungsweise im Rahmen des ProSiT Konzepts von der Klasse ProSiTEnity. Jedes Attribut, dass für ein Objekt in der Objektsicht definiert wurde, wird als ein Attribut in die Klasse mit entsprechenden set- und get-Methoden überführt.

3.6.2 Vom Transformationsmodell zu Arena

Nachfolgend werden die Anwendbarkeitsregeln sowie Transformationsregeln aufgeführt, mit denen das Transformationsmodell nach Arena überführt werden kann. Für die Transformation nach Arena sind drei Anwendbarkeitsregeln definiert. Trifft die erste Anwendbarkeitsregel $arAR_1$ zu, dann kann das Modell nicht nach Arena überführt werden. Die Anwendbarkeitsregeln $arAR_2$ und $arAR_3$ hingegen, sagen nur aus, dass das Modell nicht überführt werden sollte. Der Grund hierfür liegt in Leistungseinbußen, wenn ein Simulationslauf durchgeführt wird.

Anwendbarkeitsregel $arAR_1$ schließt das Element der terminierenden Senke aus. In Arena sind alle Marken unabhängig voneinander und können nicht gegenseitig aufeinander zugreifen. So kann eine Marke, welche die Senke erreicht, alle anderen Marken mit gleicher Seriennummer aus dem Modell entfernen. So kann aus einem Process Element die darin liegende Marke nicht entfernt werden, wodurch eine terminierende Senke nicht realisierbar ist.

Anwendbarkeitsregel $arAR_1$
If the process contains terminating sink, then the process can not be converted to Arena.

Die Anwendbarkeitsregeln $arAR_1$ und $arAR_2$ beziehen sich auf ein zusammenführendes inklusives und zusammenführendes komplexes Gateway. Diese können den Fall „first come“ aufweisen, bei dem nur die Erste ankommende Marke nach dem zusammenführenden Gateway weitergegeben wird. Dieses Verhalten ist in Arena realisierbar, jedoch durch Leistungseinbußen erkauft, die exponentiell mit der Anzahl der Marken ansteigt, die das Simulationsmodell durchlaufen. Die Anzahl der Marken zur Ermittlung dieser Abhängigkeit wurde durch ein einfaches Simulationsmodell erhoben, welches den zusammen-

führenden Oder Gateway mit „first-come“ Verhalten realisiert. Der entscheidende Modellausschnitt ist in Abbildung 45 aufgeführt.

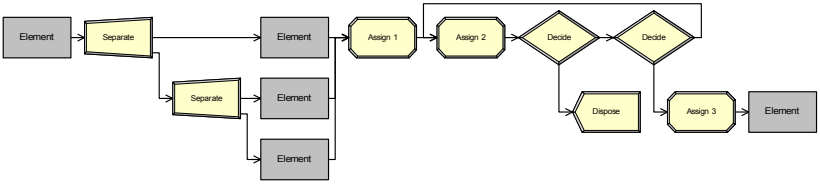


Abbildung 45: Ermittlung der If-Abfragen eines „first-come“ Oders

Die Separate Elemente repräsentieren das verzweigende inklusive Gateway. Diese sorgen dafür, dass immer alle Prozesspfade, repräsentiert durch die drei parallelen Elemente, ausgeführt werden. Damit nur das erste ankommende Element eines zusammenführenden inklusiven Gateways weitergeleitet wird, ist eine Tabelle notwendig, welche die Seriennummern der einzelnen Marken enthält. Diese wird in Arena mittels einer Variable „elements“ gepflegt, welche x-Reihen und eine Spalte umfasst. Die Anzahl der Zeilen x legt fest, wie viele Marken mit unterschiedlicher Seriennummer unterstützt werden. Hierfür ist ein fester Wert vor der Durchführung einer Simulationsstudie festzulegen. Dieser muss größer sein, als die Anzahl der Marken die das inklusive Oder Konstrukt durchlaufen, damit es zu keinem Laufzeitfehler führt. Für das Beispiel in Abbildung 45 wird der Wert auf 1000 gesetzt.

Das „Assign 1“ Element pflegt das Attribut „line“ mit dem Wert 0. Dieses ist das Attribut, welches angibt welche Zeile aus der „elements“ Variable ausgelesen werden soll. Das „Assign 2“ Element ist Teil der Schleife und erhöht das Attribut „line“ um den Wert 1, wodurch beim ersten Durchlauf die erste Zeile der Tabelle ausgelesen werden kann. Daraufhin prüft das „Decide 1“ Element das Attribut „Entity.SerialNumber“ auf Ungleichheit mit dem aktuellen Eintrag der Tabelle „elements(line, 1)“. Ist diese Bedingung nicht wahr, wird die Marke über das „Dispose“ Element aus dem Simulationsmodell entfernt. Marken, die das „Dispose“ Element erreichen, dürfen nicht in die Statistik einfließen, um die Simulationsergebnisse nicht zu verfälschen. Der semantische Hintergrund ist: Erreicht eine Marke das

„Dispose“ Element, so hat bereits eine Marke mit gleicher Seriennummer das zusammenführende inklusive Gateway passiert.

Wenn das Ergebnis des „Decide 1“ Elements wahr ist – die Seriennummern stimmt nicht übereinstimmen – wird im „Decide 2“ Element die Variable „elements“ an der Zeile „line“ und der Spalte 1 auf Ungleichheit mit dem Wert „0.0“ geprüft, welches den Initialwert für die einzelnen Zeilen darstellt. Ist dies der Fall, dann erfolgt ein Rücksprung zum „Assign 2“ Element. Semantisch bedeutet diese Prüfung, dass der aktuelle Eintrag nicht leer ist, womit die nächste Zeile geprüft wird. Ist hingegen der Eintrag in der Tabelle leer, dann wurde die letzte eingetragene Seriennummer in der Tabelle erreicht, womit noch keine Marke mit der gleichen Seriennummer das zusammenführende inklusive Gateway erreichte. Daher wird mittels des „Assign 3“ Elements in der Zeile „line“ und der Spalte 1 der Variable „elements“ die Seriennummer „Entity.SerialNumber“ eingetragen, wodurch verhindert wird, dass eine weitere Marke mit der gleichen Seriennummer weitergereicht wird.

Durch Konfiguration der Quelle wurde das Attribut der maximal zu erstellenden Marken schrittweise erhöht und die Anzahl der IF-Abfragen, wie in Tabelle 29 aufgeführt, für die beiden Decide Elemente notiert.

Tabelle 29: Maximale Anzahl an Abfragen eines „first-come“ inklusiven Oders

Anzahl Marken	Anzahl Prüfungen Decide 1	Anzahl Prüfungen Decide 2	Anzahl Prüfungen Gesamt
1	3	1	4
2	9	5	14
3	18	12	30
4	30	22	52
5	45	35	80
6	63	51	114
7	84	70	154

Aus Tabelle 29 folgt die nachfolgende Formel, welche die Summe der ausgeführten IF-Abfragen beschreibt:

$$f(n)=\sum_{i=1}^{n}(i-1)\cdot w(i)+1$$

Die Variable n gibt an, wie viele Marken das verzweigende inklusive Gateway erreichen. Mit der Variablen i wird die durchlaufende Nummer der Marken bezeichnet, die das verzweigende Gateway erreichten. Die Funktion $w(i)$ hingegen gibt an, wie viele Prozesspfade durch das inklusive Gateway für die jeweilige Marke ausgeführt werden. Bezogen auf die Werte der Tabelle 29 ist $w(i)=3$. Hinter dieser Funktion liegt die Verteilung, die aus dem verzweigenden Oder Gateway folgt.

Wie aus Tabelle 29 hervorgeht, liegt ein exponentielles Wachstum der IF-Anfragen vor. Im schlechtesten Fall erfolgen bei 100 Marken 30.100 IF-Abfragen, im besten Fall, wenn jeweils nur ein Pfad nach dem inklusiven Oder ausgelöst wird, 10.100 IF-Abfragen, welche die Dauer der Durchführung einer Simulationsstudie erhöhen. Aufgrund dieses Zusammenhangs folgt die Konklusion mittels der Vorbereitungsregeln $arAR_2$ und $arAR_3$, dass ein zusammenführendes inklusives „first-come“ Oder nicht simuliert werden sollte, die Anwendung des Konstrukts jedoch nicht ausschließen.

Anwendbarkeitsregel $arAR_2$
If the process contains first-come joining inclusive gateway, then the process should not be converted to Arena.

Semantisch gibt es zwischen dem zusammenführenden inklusiven „first-come“ Gateway und dem zusammenführenden komplexen „first-come“ Gateway keinen Unterschied. Lediglich die verzweigenden Gateways unterscheiden sich, in dem das inklusive Gateway die auszuführenden Prozesspfade mittels einer Wahrscheinlichkeit bestimmt und das komplexe Gateway dies basierend auf Attributen einer Marke vollzieht. Dementsprechend ist die Anwendbarkeitsregel $arAR_3$ identisch formuliert, wie Anwendbarkeitsregel $arAR_2$.

Anwendbarkeitsregel $arAR_3$
If the process contains first-come joining complex gateway, then the process should not be converted to Arena.

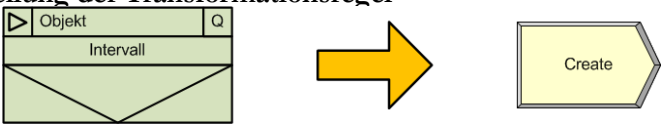
Transformationsregel – arTR (Arena Transformation Rule)

Mit den aufgeführten Transformationsregeln erfolgt die Überführung des Transformationsmodells in die Simulationsumgebung Arena.

Transformationsregel $arTR_1$
If the element is a source, then it will be a create element.

Die Transformationsregel $arTR_1$ überführt eine Quelle in das äquivalente Element in Arena. Das Create Element erzeugt die Marken, die das Simulationsmodell durchlaufen. Hierfür müssen die Attribute übernommen werden. Das Objekt der Quelle wird als „Entity Type" im Create Element verwendet und die Quantität entspricht dem Attribut „Entities per Arrival". Mittels der drei Attribute „Type", „Expression" und „Units" kann letztlich die Verteilung, die in der Quelle verwendet wird, überführt werden. Verschiedene Verteilungen können mit dem Wert „Expression" im Attribut „Typ" hinterlegt werden. Im Attribut „Expression" kann daraufhin eine äquivalente Verteilungsfunktion ausgewählt werden.

Darstellung der Transformationsregel



Transformationsregel $arTR_2$
If the element is a sink, then it will be a dispose element.

Transformationsregel $arTR_2$ überführt eine Senke in ein Dispose Element. Das Dispose Element entfernt Marken aus dem Simulationsmodell, wodurch dieses das gleiche semantische Verhalten aufweist. Im Gegensatz zur Quelle muss nur ein Attribute gepflegt werden, welches jedoch nicht abhängig von der Senke des Transformationsmodells ist. Das Attribute „Record Entity Statistics"

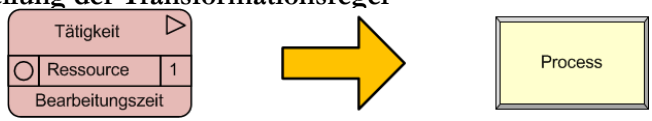
muss aktiviert werden, damit Marken, welche das Dispose Element erreichen, in der Auswertungsstatistik von Arena berücksichtigt werden.

Darstellung der Transformationsregel

Transformationsregel $arTR_3$
If an activity has not a marker or has a combined marker, then it will be a process element. The processing time of the activity will be converted to the process element and if resources are not bound, then they will be added as a resource to the process element.

Die Transformationsregel $arTR_3$ betrachtet Aktivitäten ohne Marker und kombinierte Aktivitäten, die durch einen entsprechenden Marker gekennzeichnet sind. Eine Aktivität entspricht in Arena dem Process Element. Die Bearbeitungszeit der Aktivität wird, ähnlich wie bei der Quelle in der Transformationsregel $arTR_1$, mittels der drei Attribute „Delay Type“, „Units“ sowie „Expression“ überführt. Zur Überführung der Ressourcen, die in der Aktivität definiert sind, sind die Ressourcenbindungs und -freigabe Elemente zu berücksichtigen. Wenn eine Ressource durch eine Ressourcenbindung für den Prozesspfad gebunden ist, dann wird diese nicht dem Process Element hinzugefügt. Sind alle Ressourcen gebunden, dann wird das Attribut „Action“ mit dem Wert „Delay“ versehen, da keine weiteren Ressourcen benötigt werden. Ist jedoch mindestens eine weitere Ressource notwendig, dann wird der Wert „Seize Delay Release“ verwendet und die Ressource wird dem Process Element hinzugefügt.

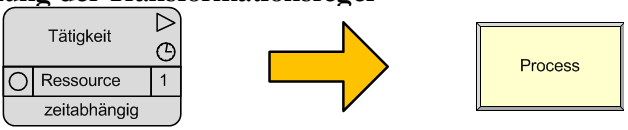
Darstellung der Transformationsregel



Transformationsregel $arTR_4$
If an activity has a time dependent Marker, then is will be a process element without a processing time.

Die Transformationsregel $arTR_4$ überführt zeitabhängige Aktivitäten in Process Elemente. Das Process Element erhält jedoch keine Bearbeitungszeit, da diese von anderen parallel ausgeführten Aktivitäten abhängig ist. Die Realisierung dieses Aspekts ist durch die Normalisierungsregel sNR_8 berücksichtigt. Wenn diese Normalisierungsregel angewendet wurde, sind alle Ressourcen der Aktivität bereits gebunden. Das Attribut „Action“ muss mit dem Wert „Delay“ versehen werden. Damit die Aktivität jedoch zeitlich von einer parallelen Aktivität abhängig ist, muss als „Delay Type“ der Wert „Const“ angegeben werden und das Attribut „Value“ auf 0 festgelegt sein. Aus dem Bericht, den Arena nach einem Simulationslauf erzeugt, ist durch diese Umsetzung keine Bearbeitungszeit für die Aktivität entnehmbar. Diese kann jedoch aus der parallel ausgeführten Aktivität entnommen werden. Würde die zeitabhängige Aktivität nicht überführt, dann könnte diese zu Verunsicherung bei der Evaluation des Simulationsmodells durch einen Modellierer führen, weshalb das Element berücksichtigt wird.

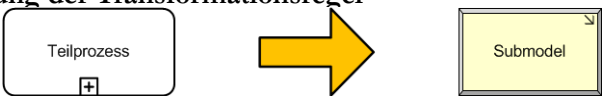
Darstellung der Transformationsregel



Transformationsregel $arTR_5$
If the element is a sub process, then it will be a submodel element.

Die Transformationsregel $arTR_5$ betrifft den Teilprozess, der in ein Submodel Element in Arena überführt wird. Streng genommen ist dies kein separates Element, sondern ein Process Element, bei dem das Attribut „Type“ auf den Wert „Submodel“ festgelegt wurde. Dadurch ändert sich jedoch auch die grafische Repräsentation des Elements, womit von einem separaten Element gesprochen wird. Bei der Überführung der Elemente des Teilprozesses ist darauf zu achten, dass diese in das korrekte Submodel Element überführt werden. Ein Submodel Element hat eine Einschränkung, es gibt nur einen Eingang und einen Ausgang zum umgebenden Modell.

Darstellung der Transformationsregel



Transformationsregel $arTR_6$
If the element is a resource binding, then it will be a seize element.

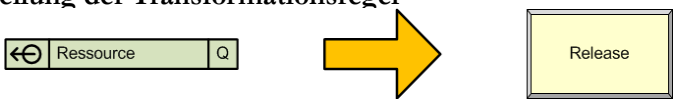
Transformationsregel $arTR_6$ überführt eine Ressourcenbindung in ein Seize Element. Das Verhalten beider Elemente ist identisch, da die Ressourcenbindung aus dem Seize Element abgeleitet wurde. Beide Elemente entnehmen eine Ressource aus dem Pool der frei zur Verfügung stehenden Ressourcen. Stehen nicht genügend freie Ressourcen zur Verfügung pausiert die Marke, bis die Ressourcen für die Ressourcenbindung zur Verfügung stehen. In Arena müssen hierfür lediglich die Ressourcen in „Resources“ Liste des Seize Elements mit ihrer entsprechenden Quantität übernommen werden.

Darstellung der Transformationsregel

Transformationsregel $arTR_7$
If the element is a resource release, then it will be a release element.

Transformationsregel $arTR_7$ überführt das Gegenstück der Ressourcenbindung, die Ressourcenfreigabe, in ein Release Element. Die Ausführungen zur Transformationsregel $arTR_6$ gelten analog für die Transformationsregel $arTR_7$. Eine Marke hat aber keine Verweilzeit in einer Ressourcenfreigabe. Erreicht eine Marke das Element, dann werden die in der „Ressource“ Liste aufgeführten Ressourcen in der angegebenen Menge freigegeben. Zu beachten ist: Wenn mehr Ressourcen freigegeben werden als reserviert sind, führt dies zu einem Laufzeitfehler.

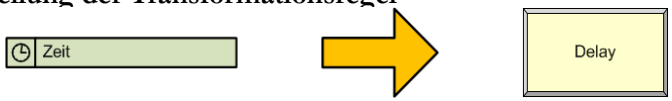
Darstellung der Transformationsregel



Transformationsregel $arTR_8$
If the element is a delay, then it will be a delay element.

Transformationsregel $arTR_8$ überführt die Verzögerung des ProSiT Ablaufdiagramms in ein Delay Element in Arena. Beide Elemente verzögern eine Marke. Die Zeit der Verzögerung wird in das Attribute „Delay Time“ überführt, unter Berücksichtigung der Zeiteinheit „Units“.

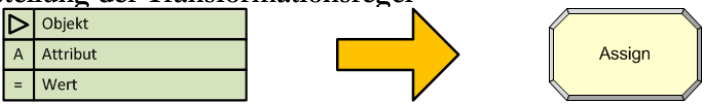
Darstellung der Transformationsregel



Transformationsregel $arTR_9$
If the element is an attribute set, then it will be an assign element.

Transformationsregel $arTR_9$ transformiert eine Attributsfestlegung in ein Assign Element. Während das Assign Element die Festlegung von mehr als einem Attribut erlaubt, kann die Attributsfestlegung nur den Wert eines Attributes ändern. Die Transformationsregel ist davon jedoch nicht betroffen, da die Einschränkung beim Quellmodell vorliegt. Im Assign Element wird der „Type“ auf den Wert „Attribute“ festgelegt und im Feld „Attribute Name“ der Name des Attributs eingetragen. Über das Attribute „New Value“ wird der Wert übernommen, der beim Assign Element auch eine Verteilung sein kann.

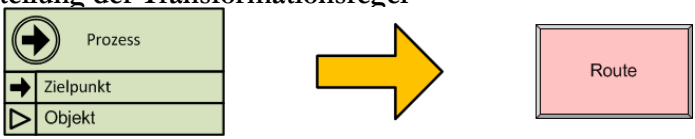
Darstellung der Transformationsregel



Transformationsregel $arTR_{10}$
If the element is a jump point, then it will be a route element.

Transformationsregel $arTR_{10}$ überführt den Sprungpunkt in ein Route Element von Arena. Eine Übertragungszeit, welche das Route Element unterstützt, wird beim Sprungpunkt nicht berücksichtigt, weshalb die „Route Time“ auf 0 gesetzt wird. Als „Destination Type“ wird der Wert „Station“ verwendet, welche den Zielpunkt der Weiterleitung darstellt. Im Attribut „Station Name“ wird der Zielpunkt aus dem Sprungpunkt übernommen. Das Objekt des Sprungpunkts ist im ProSiT Ablaufdiagramm rein informativ und wird daher für die Transformationsregel $arTR_{10}$ nicht benötigt.

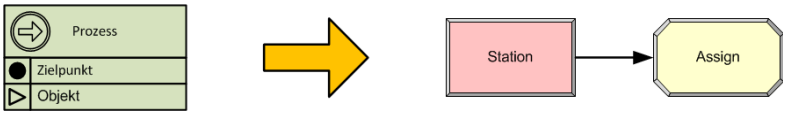
Darstellung der Transformationsregel



Transformationsregel $arTR_{11}$
If the element is a target point, then it will be a station element, followed by an assign element, that changes the entity type.

Transformationsregel $arTR_{11}$ überführt das Gegenstück des Sprungpunktes, den Zielpunkt, in ein Station Element. Der Zielpunkt des Sprungpunktes wird für das Attribute „Station Name“ verwendet, womit sichergestellt wird, dass das Simulationsmodell einen entsprechenden Zielpunkt für ein Route Element aufweist. Damit das Station Element entsprechend verwendet wird, muss hierfür das Attribut „Station Type“ auf den Wert „Station“ festgelegt werden. Da das Zielpunktelement keine Verknüpfung zu möglichen Sprungpunkten aufweist, kann nicht geprüft werden, ob sich das Objekt zwischen dem Sprungpunkt und dem Zielpunkt ändert. Daher wird der Station ein Assign Element als Nachfolger eingefügt, welches das Attribut „Entity.Type“ mit dem Objekt des Zielpunktes festlegt.

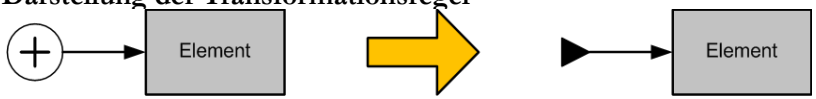
Darstellung der Transformationsregel



Transformationsregel $arTR_{12}$
If the element is a sub process start event, then the successor will be get the entry as its predecessor.

Transformationsregel $arTR_{12}$ überführt das Teilprozessstartereignis nach Arena. Im ProSiT Ablaufdiagramm stellt das Element den Startpunkt für den Teilprozess dar. In Arena wird dieser Startpunkt über ein „Entry", ein Dreieck im Teilprozess, repräsentiert. Für die Verknüpfung dieses „Entry" Elements erhält der Nachfolger des Teilprozessstartereignisses dieses als Vorgänger. Durch diese Verknüpfung werden Marken, die den Teilprozess betreten, an dieses erste Element geleitet.

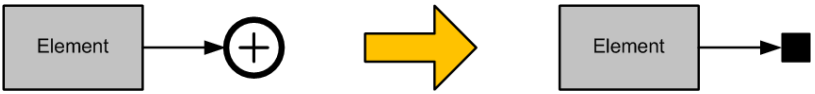
Darstellung der Transformationsregel



Transformationsregel $arTR_{13}$
If the element is a sub process end event, then the predecessor will get the exit as its successor.

Transformationsregel $arTR_{13}$ ist das Gegenstück zur Transformationsregel $arTR_{12}$. Entsprechend den Ausführungen zum Teilprozessstartereignis wird der Vorgänger des Teilprozessendereignisses mit dem „Exit" des Teilprozesses in einer Nachfolgerbeziehung verknüpft.

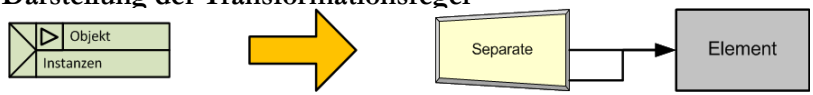
Darstellung der Transformationsregel



Transformationsregel $arTR_{14}$
If the element is a splitting instance, then it will be a separate element.

Transformationsregel $arTR_{14}$ wandelt die verzweigende Instanziierung in ein Separate Element. Damit das Separate Element das gleiche Verhalten wie die verzweigende Instanziierung aufweist, muss das Attribut „Type“ mit dem Wert „Duplicate Original“ versehen werden. Das Attribut „# of Duplicates“ erhält die Anzahl $n-1$, wobei n für die Anzahl steht, die im Feld Instanzen der verzweigenden Instanziierung gepflegt sind. Neben den Attributen müssen die Ausgänge „Original“ und „Duplicate“ mit dem Nachfolger der verzweigenden Instanziierung verknüpft werden. Das Separate Element leitet damit die Originalmarke weiter sowie $n-1$ Duplikate, wodurch die geforderte Anzahl an Instanzen erzeugt wird.

Darstellung der Transformationsregel

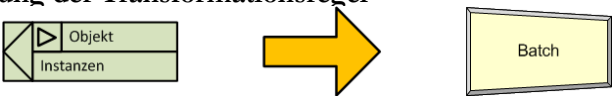


Transformationsregel $arTR_{15}$
If the element is a merging instance, then it will be a batch element.

Transformationsregel $arTR_{15}$ führt mehrere Instanzen einer Marke zusammen. Hierfür wird die zusammenführende Instanziierung in ein Batch Element überführt. Diese unterstellt, dass nur Marken zusammengeführt werden können, die über die gleiche Seriennummer verfügen, womit das Element eine vorangehende verzweigende Instanziierung voraussetzt. Zur Realisierung dieses Verhaltens wird das Attribut „Type“ auf den Wert „Permanent“ festgelegt, die Regel des

Zusammenführend („Rule“) auf „By Attribute“ und der „Attribut Name“ auf „Entity.SerialNumber“. Als weiteres Attribut muss festgelegt werden, wie viele Instanzen zusammengeführt werden sollen. Dies erfolgt über das Attribut „Batch Size“, welches den Wert der Instanzen der zusammenführenden Instanziierung erhält.

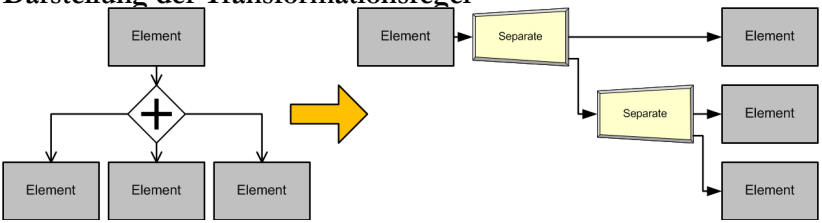
Darstellung der Transformationsregel



Transformationsregel $arTR_{16}$
If a parallel gateway has one predecessor and more than one successor then it will be a series of separate elements in the quantity of the number of the successors minus one.

Das Element, welches eine parallele Ausführung erlaubt, ist das Separate Element. Das parallele Gateway wird durch die Transformationsregel $arTR_{16}$ in dieses überführt. Das verzweigende parallele Gateway kann jedoch nicht in ein Element in Arena überführt werden, wenn mehr als zwei Nachfolger vorhanden sind. Das Separate Element verfügt nur über zwei Ausgänge. Prinzipiell werden alle Duplikate über den „Duplicate“ Ausgang weitergeleitet. Sind drei parallele Prozesspfade durchzuführen, können die Separate Elemente geschachtelt werden. Für n parallele Prozesspfade sind $n-1$ Separate Elemente notwendig, die jeweils ein Duplikat („# of Duplicates“) erzeugen.

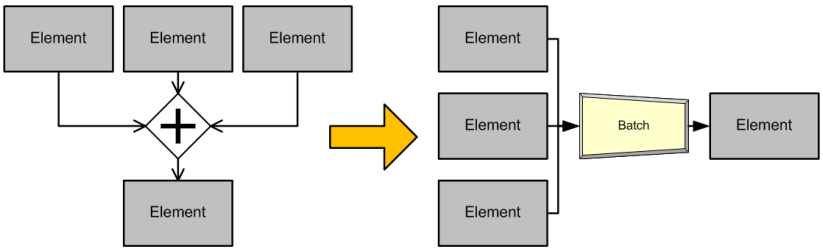
Darstellung der Transformationsregel



Transformationsregel $arTR_{17}$
If a parallel gateway has more than one predecessor and one successor, then it will be a batch element.

Transformationsregel $arTR_{17}$ entspricht hinsichtlich ihrer Überführung der Transformationsregel $arTR_{15}$. Aus einem zusammenführenden parallelen Gateway wird ein Batch Element erzeugt. Dieses synchronisiert alle vorangehenden Prozesspfade und sendet eine Marke weiter. Da eine Synchronisation nur über die Seriennummer der Marke vollzogen wird, benötigt ein Batch Element ein verzweigendes paralleles Gateway. Damit das Batch Element diese Aufgabe erfüllen kann, muss das Attribute „Type“ den Wert „Permanent“ erhalten und die Regel des Zusammenführens („Rule“) auf „By Attribute“ eingestellt werden. Damit Seriennummern abgeglichen werden, ist das Feld „Attribute Name“ „Entity.SerialNumber“ notwendig. Die Anzahl der Marken, die synchronisiert werden sollen, ist aus dem Kontext zu entnehmen. So wird die „Batch Size“ durch die Anzahl der Vorgänger des zusammenführenden parallelen Gateways bestimmt.

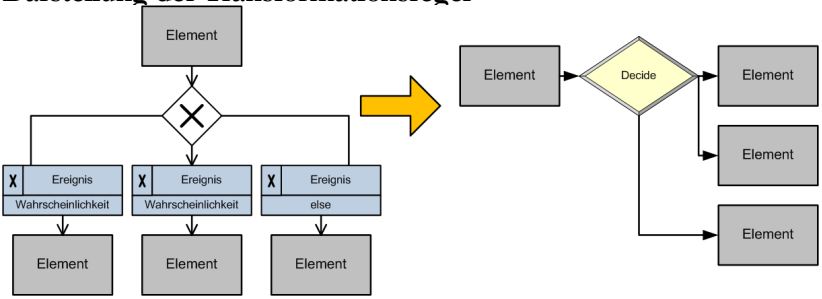
Darstellung der Transformationsregel



Transformationsregel arTR$_{18}$
If an exclusive gateway has one predecessor and more than one successor, then it will be a decide element and the probabilities of the exclusive keys that are the successors of the gateway will be added to the decide element.

Transformationsregel arTR$_{18}$ überführt ein verzweigendes exklusives Gateway in ein Decide Element. Damit dieses wahrscheinlichkeitsbasiert ist, muss das Attribut „Type“ auf „N-way by Chance“ eingestellt werden. In die Liste „Percentages“ können daraufhin die Wahrscheinlichkeiten der nachfolgenden exklusiven Schlüssel eingetragen werden. Zu beachten ist: Nur $n-1$ Wahrscheinlichkeiten können eingetragen werden. Der letzte Prozesspfad wird über den else-Ausgang verbunden. Wird ein exklusiver else-Schlüssel verwendet, dann wird dieses Element über den else-Ausgang angesteuert.

Darstellung der Transformationsregel

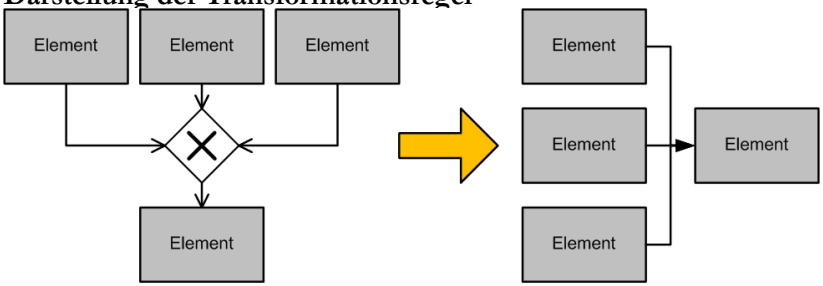


Transformationsregel arTR$_{19}$
If an exclusive gateway has more than one predecessor and one successor, then it will be removed and the predecessors of the gateway will get the successor as their successor.

Im Gegensatz zu anderen Transformationsregeln entfernt die Transformationsregel arTR$_{19}$ das zusammenführende exklusive Gateway aus dem Modell. In Arena ist kein Element vorhanden, welches mehrere mögliche Prozesspfade zusammenführt. Stattdessen kann ein Element über mehrere Vorgänger verfügen, die in einer exklusiven Oder

Beziehung zueinander stehen. Daher erhalten die Vorgänger des Gateways dessen Nachfolger als Nachfolger.

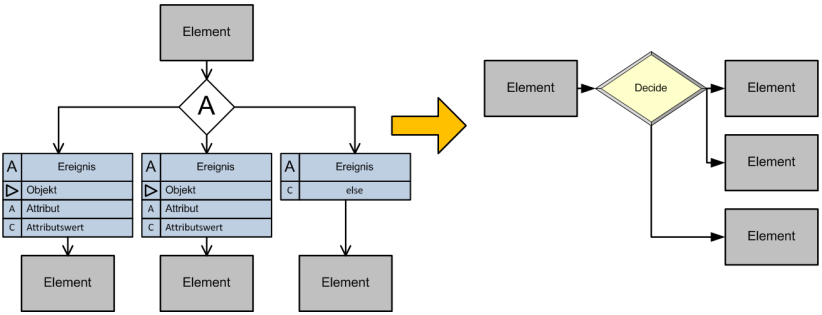
Darstellung der Transformationsregel



Transformationsregel $arTR_{20}$
If an attribute-based gateway has one predecessor and more than one successor, then it will be a decide element and the attributes of the attribute-based keys that are the successors of the gateway will be added to the decide element.

Transformationsregel $arTR_{20}$ überführt analog zur Transformationsregel $arTR_{18}$ ein verzweigendes attributsbasiertes Gateway in ein Decide Element. Das Attribut „Type“ wird mit „N-way by Condition“ versehen. Hierdurch können in die „Conditions“ Liste die Prüfungen nach einem Attribut eingetragen werden. Für die Überführung ist ein else-Schlüssel zwingend notwendig, da anderweitig nicht sichergestellt werden kann, dass die Prüfung nach den Attributen jeden möglichen Fall abdeckt. Dies muss in der Transformationsregel arTR20 jedoch nicht speziell berücksichtigt werden, da dies durch die Normalisierungsregel sNR_{11} abgedeckt wird. Der else-Schlüssel selbst wird über den Else-Ausgang des Decide Elements realisiert.

Darstellung der Transformationsregel



Transformationsregel $arTR_{21}$
If an attribute-based gateway has more than one predecessor and one successor, then it will be removed and the predecessors of the gateway will get the successor as their successor.

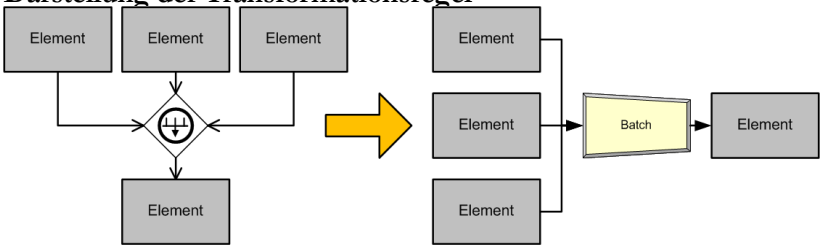
Transformationsregel $arTR_{21}$ ist identisch mit der Transformationsregel $arTR_{19}$, lediglich das Gateway ist ein zusammenführendes attributsbasiertes Gateway. Das Verhalten ist jedoch identisch mit dem des zusammenführenden exklusiven Gateways, sodass auf Transformationsregel $arTR_{19}$ verwiesen wird.

Transformationsregel $arTR_{22}$
If an inclusive gateway or a wait-for-all inclusive gateway has more than one predecessor and one successor, then it will be a batch element.

Transformationsregel $arTR_{22}$ überführt ein inklusives „wait-for-all" Gateway in ein Batch Element. Ein inklusives Gateway, welches nicht speziell markiert ist und mehr als einen Vorgänger und einen Nachfolger hat, wird ebenfalls als ein inklusives „wait-for-all" Gateway aufgefasst. Das Batch Element ist für die Synchronisation der eingehenden Prozesspfade zuständig. Die Anzahl der Vorgänger wird hierfür in das Attribut „Batch Size" überführt. Als Regel („Rule") wird „by Attribute" ausgewählt mit dem Attribut „Entity.SerialNumber".

Um ein wait-for-all Konstrukt korrekt abzubilden, ist neben der Transformationsregel $arTR_{22}$ und $arTR_{25}$ notwendig.

Darstellung der Transformationsregel



Transformationsregel $arTR_{23}$
If a first-come inclusive gateway has more than one predecessor and one successor, then it will we be a sequence of two assign elements and two decide elements. The second successor of the first decide element will be a dispose element. The first successor of the second decide element will the the second assign element and the second successor will be an additional assign element that has the the successor of the gateway as its successor.

Die Transformationsregel $arTR_{23}$ überführt das semantische Verhalten des zusammenführenden „first-come" inklusiven Gateways nach Arena. Zur Verdeutlichung des notwendigen Ablaufes zur Realisierung dieses Gateways wird auf die dargestellte Abbildung verwiesen. Vor der Erläuterung werden die Attribute der einzelnen Elemente in Tabelle 30 aufgeführt.

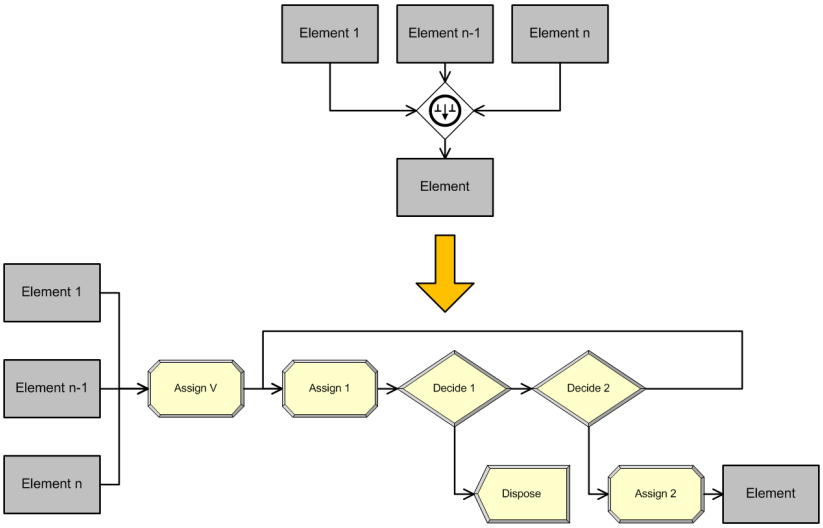
Tabelle 30: Parameter der Elemente der Transformationsregel $arTR_{23}$

Element	Assignments/Decide
Assign V	Attribute, pos, 0
Assign 1	Attribute, pos, pos +1
Assign 2	Variable Array (2D), firstcome_element, pos, 1, Entity.Serialnumber
Decide 1	If Attribute Entity.SerialNumber <> firstcome_element(pos, 1)
Decide 2	If Variable Array (2D) firstcome_element(pos, 1) <> 0.0

Das dargestellte Konstrukt benötigt eine zweidimensionale Variable „firstcome_element“. Dieser verfügt über eine Spalte und über n Zeilen. In dieser Variablen werden die Seriennummern gespeichert, die das inklusive Gateway bereits passierten.

Das Assign V Element legt zunächst fest, dass die Zeile 0 der Variable auszulesen ist. Diese wird im nächsten Schritt um eins erhöht, um die erste Zeile auszulesen. Das Decide 1 Element prüft daraufhin, ob die Seriennummer in dieser Zeile der Seriennummer der Marke nicht entspricht. Wenn die Seriennummern übereinstimmen, dann bedeutet dies, dass eine Marke mit der gleichen Seriennummer bereits eingetragen wurde. Die geprüfte Marke wird daher über das Dispose Element aus dem Simulationsmodell entfernt. Trifft diese else-Bedingung nicht zu, dann erfolgt eine weitere Prüfung mit dem Decide 2 Element. Diese prüft, ob der aktuelle Eintrag der Tabelle dem Initialwert 0.0 nicht entspricht. Ist diese Bedingung wahr, dann erfolgt ein Rücksprung zum Assign 1 Element, mit dem die nächste auszulösende Zeile festgelegt wird. Trifft die Bedingung nicht zu, dann hat noch keine Marke mit der gleichen Seriennummer das inklusive Gateway passiert. Daher wird an der aktuellen Zeile, die den Wert 0.0 enthält, die Seriennummer der Marke eingetragen, damit keine weitere Marke mit der gleichen Seriennummer passieren kann. Für die korrekte Funktion dieses Ablaufs bedarf es der Transformationsregel $arTR_{26}$, welche den entsprechenden Verzweigungsteil nach Arena überführt.

Darstellung der Transformationsregel



Transformationsregel $arTR_{24}$
If an every-time inclusive gateway has more than one predecessor and one successor, then it will be removed and the predecessors of the gateway will get the successor as their successor.

Transformationsregel $arTR_{22}$ arbeitet analog zu den Transformationsregeln $arTR_{19}$ und $arTR_{21}$.Das „every-time“ inklusive Gateway wird aus dem Modell entfernt, und die Vorgänger werden direkt mit dem Nachfolger verknüpft. Hiermit wird jeder ausgeführte Prozesspfad ohne Synchronisation weitergeleitet. Für eine semantisch korrekte Anwendung ist es jedoch notwendig, dass die Transformationsregeln $arTR_{24}$ für das verzweigende inklusive Gateway Anwendung findet.

Transformationsregel $arTR_{25}$
If an inclusive gateway or a wait-for-all inclusive gateway has one predecessor and more than one successor, then it will be a construct of several elements. First it will become an initial assign element. Let n+1 be the number of paths that follow the gateway. Every path until n-1 will get a sequence of an assign element, a decide element, an assign element and a separate element and a second assign element that is the successor of the else path of the decide element. One successor of the first separate element is the successor of the inclusive gateway. The second successor of the first separate element and the first successor of the second separate element is the first element of the next sequence i+1 and the second successor of the second separate element will be the combining wait-for-all inclusive gateway. The n element is different to the previous elements and only consists of the first assign, the separate element. An else path will be a separate element and a decide element that is the successor of the separate else element. The first successor of the element will be the successor of the else key and the second successor will be the combining gateway.

Transformationsregel $arTR_{25}$ stellt die komplexeste Transformationsregel dar, um ein Element des ProSiT Ablaufdiagramms nach Arena zu überführen. Die Regel geht davon aus, dass ein inclusives „wait-for-all" Gateway $n+1$ Nachfolger hat. Im dargestellten Beispiel sind dies 4 Nachfolger, wobei ein Prozesspfad durch einen else-Schlüssel angesteuert wird und durch die 1 bei der Anzahl der Nachfolger gekennzeichnet wird. Entsprechend ist die Transformationsregel dreigeteilt. Eine detaillierte Übersicht über die festzulegenden Attribute ist in Tabelle 32 aufgeführt.

Die erste Transformation erfolgt für die Elemente 1 bis $n-1$. Zunächst erfolgt mit dem Element Assign V eine Initialisierung. Für dieses ist eine zweidimensionale Variable „waitforall" definiert. Diese enthält codiert die Wahrscheinlichkeiten für die Ausführung der einzelnen Prozesspfade. Hierbei werden Wahrscheinlichkeiten für Prozesspaare vergeben. Ein entsprechendes Beispiel ist in Tabelle 31 aufgeführt.

Tabelle 31: Wahrscheinlichkeitstabelle für verzweigende inklusive Gateways

Pfade	Kumulierte Wahrscheinlichkeit
100	0,1
200	0,2
300	0,3
120	0,45
130	0,6
230	0,75
123	0,9
000	1

Damit das durch die Transformationsregel erzeugte Konstrukt funktionsfähig ist, müssen alle Zeichenketten der ersten Spalte in der Anzahl der Zeichen die Anzahl der maximal zeitlich ausführbaren Pfade umfassen sowie aufsteigend geordnet sein. Wenn durch die Wahrscheinlichkeitsverteilung die erste Zeile ausgeführt wird, dann wird nur Pfad 1 ausgeführt. Die 0 dient hierbei als Platzhalter. Eine Ausnahme davon bildet die letzte Zeile. Diese besteht aus drei mal der „0“ und kennzeichnet den Pfad mit dem else-Schlüssel. Mit dem ersten Attribut des Assign V Elements erfolgt basierend auf dieser Tabelle mittels einer diskreten Wahrscheinlichkeit (DISC) die Festlegung der auszuführenden Prozesspfade (path). Ebenfalls wird das Attribute „pos“ mit dem Wert 1 versehen. Dieses gibt an, welche Stelle der Zeichenkette (beispielsweise „120“) als nächstes ausgelesen werden soll.

Dieses Auslesen erfolgt beim ersten Assign Element, bei dem eine Ziffer ausgelesen wird und dem Attribut „process“ zugewiesen wird. Die Prüfung erfolgt im nachfolgenden Decide Element, in dem beim ersten Prozesspfad der Wert „1“ gegengeprüft wird. Ist diese Prüfung wahr, dann wird mit dem zweiten Assign Element der Wert des Attributs „pos“ um „1“ erhöht. Das erste Separate Element leitet daraufhin eine Marke an den auszuführenden Prozesspfad weiter, sowie eine Kopie an die nächste Elementkette, im dargestellten Beispiel die

$n-1$ Elementkette. Ist die Prüfung im Decide Element jedoch nicht wahr, dann wird die Marke an ein zweites Separate Element weitergeleitet. Diese leitet das Original an das erste Element der nächsten Elementkette weiter, sowie die Kopie an das zusammenführende inklusive wait-for-all Gateway. Zu beachten ist hierbei, dass die auszulesende Position in diesem Fall nicht erhöht wird.

Die zweite Überführung, welche die Regel ermöglicht, betrifft das Element n. Diese ließt mit dem „Assign n" Element den auszuführenden Prozesspfad aus und prüft diesen im anschließenden Decide Element. Ist die Prüfung wahr, dann wird der Prozesspfad n ausgeführt. Der zweite Ausgang des Decide Elements ist abhängig von dem Vorhandensein eines else-Pfades. Ist kein else-Pfad vorhanden, dann wird die Marke an das zusammenführende Gateway weitergeleitet. Ist jedoch ein else-Pfad vorhanden, dann wird die Marke entsprechend an diesen weitergeleitet.

Dieser stellt die dritte Prüfung der Regel dar. Der else Pfad beginnt mit einem Separate Element, welche die Marke zum einen zum zusammenführenden Gateway weiterleitet, zum anderen an ein Decide Element. Dieses prüft ob das Attribut „path" die Zeichenkette „000" enthält. Ist diese Prüfung wahr, wird der else-Pfad ausgeführt und wenn diese Prüfung falsch ist, wird die Marke ebenfalls an das zusammenführende Gateway weitergeleitet.

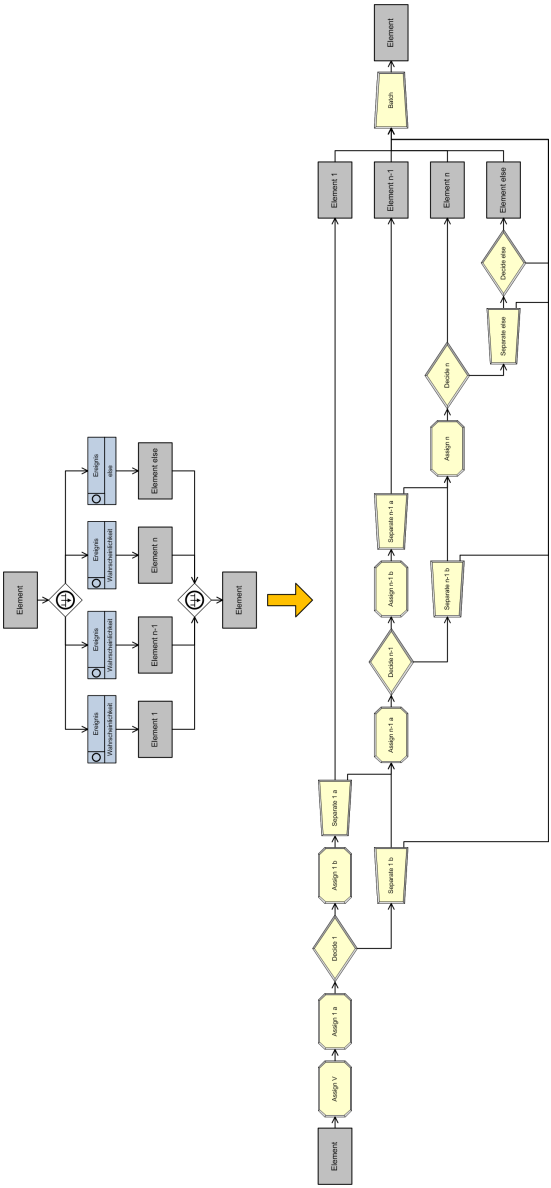
Die drei Separate Elemente (1b , $n-1\text{b}$ und $else$) sowie das Decide else Element sorgen dafür, dass am zusammenführenden Gateway immer vier Marken ankommen und bei diesem eine entsprechende Synchronisierung erfolgen kann.

Tabelle 32: Parameter der Elemente der Transformationsregel $arTR_{25}$

Element	Assignments/Decide
Assign V	Attribute, path, DISC(waitforall(1,2), waitforall(1,1), waitforall(2,2), waitforall(2,1), waitforall(3,2), waitforall(3,1), waitforall(4,2), waitforall(4,1), waitforall(5,2), waitforall(5,1), waitforall(6,2), waitforall(6,1), waitforall(7,2), waitforall(7,1), waitforall(8,2), waitforall(8,1)) Attribute, pos, 1
Assign 1 a	Attribute, process, Val(Mid(Str(path), pos, 1))
Assign 1 b	Attribute, pos, pos + 1
Decide 1	If Attribute process == 1
Assign n-1 a	Attribute, process, Val(Mid(Str(path), pos, 1))
Assign n-1 b	Attribute, pos, pos + 1
Decide n-1	If Attribute process == 2
Assign n	Attribute, process, Val(Mid(Str(path), pos, 1))
Decide n	If Attribute process == 3
Decide else	If Attribute path == 000

Im dargestellten Beispiel ist neben dem verzweigenden inklusiven wait-for-all Gateway auch das entsprechende zusammenführende Gateway aufgeführt. Wenn die Transformationsregel $arTR_{25}$ angewendet wird, dann sollte die entsprechende Transformationsregel $arTR_{22}$ angewendet werden, damit ein konsistentes Simulationsmodell vorliegt.

Darstellung der Transformationsregel



Transformationsregel $arTR_{26}$
If a first-come inclusive gateway has one predecessor and more than one successor, then it will be a construct of several elements. First it will become an initial assign element. Let n+1 be the number of paths that follow the gateway. Every path until n-1 will get a sequence of an assign element, a decide element, an assign element and a separate element. The second successor of the separate element and the second successor of the decide element is the first element of the next sequence i+1. The n element is different to the previous elements and only consists of the first assign, the separate element. An else path will be a decide element that is the successor of the decide n element. The first successor of the else decide element will be the successor of the else key and the second successor will be a drain.

Transformationsregel $arTR_{26}$ entspricht in groben Zügen der Transformationsregel $arTR_{25}$, mit dem Unterschied, dass kein zweites Separate Element verwendet wird, beziehungsweise beim else-Zweig nicht vorhanden ist. Dies folgt daraus, dass jeweils eine Marke in den ausgeführten Prozesspfaden nach dem zusammenführenden Gateway weitergeleitet werden und keine Synchronisation erfolgt. Die in Tabelle 33 aufgeführten Parameter entsprechen den Parametern der Tabelle 32. Ebenfalls ist eine zweidimensionale Variable notwendig, welche die Wahrscheinlichkeiten für die auszuführenden Prozesspfade beinhaltet, die den Angaben in Tabelle 31 entspricht.

Tabelle 33: Parameter der Elemente der Transformationsregel arTR$_{26}$

Element	Assignments/Decide
Assign V	Attribute, path, DISC(fristcome(1,2), fristcome(1,1), fristcome(2,2), fristcome(2,1), fristcome(3,2), fristcome(3,1), fristcome(4,2), fristcome(4,1), fristcome(5,2), fristcome(5,1), fristcome(6,2), fristcome(6,1), fristcome(7,2), fristcome(7,1), fristcome(8,2), fristcome(8,1)) Attribute, pos, 1
Assign 1 a	Attribute, process, Val(Mid(Str(path), pos, 1))
Assign 1 b	Attribute, pos, pos + 1
Decide 1	If Attribute process == 1
Assign n-1 a	Attribute, process, Val(Mid(Str(path), pos, 1))
Assign n-1 b	Attribute, pos, pos + 1
Decide n-1	If Attribute process == 2
Assign n	Attribute, process, Val(Mid(Str(path), pos, 1))
Decide n	If Attribute process == 3
Decide else	If Attribute path == 000

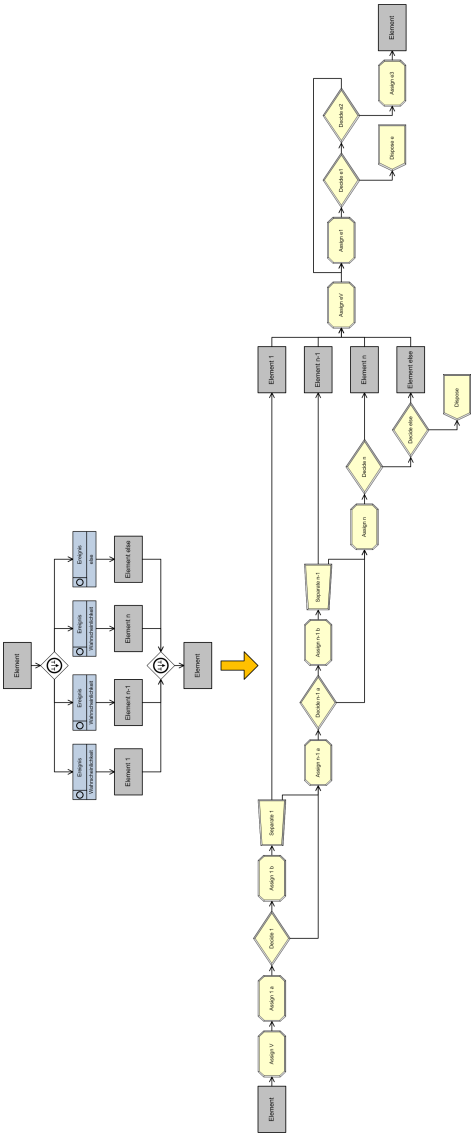
Die Decide Elemente haben so nicht als zweiten Nachfolger ein Separate Element, sondern verweisen direkt auf die nächste Elementen-Sequenz. Das am tiefsten verschachtelte Decide Element hat hierbei als zweiten Nachfolger eine Senke. Dieses entfernt die, durch die verbleibenden Separate Elemente, erzeugte Kopie, wenn diese nicht für die Ausführung des letzten Prozesspfades benötigt werden.

Das Assign V Element ist für die Initialisierung des inklusiven Operators zuständig. In diesem wird die Zeichenkette mit den auszuführenden Prozesspfaden „path“ erzeugt und die auslesende Position („pos“) mit dem Wert 1 versehen. Das Assign a Element liest daraufhin aus der Zeichenkette das erste Zeichen aus („process“), welches in dem anschließend folgenden Decide Element geprüft wird. Entspricht dieses Attribut nicht der Nummer des Prozesspfades, dann

wird die Marke an die nächste Elementsequence weitergegeben. Soll jedoch der Prozesspfad ausgeführt werden, wird anschließend im Assign b Element die auszulesende Position um den Wert 1 erhöht. Ein Separate Element sorgt darauf hin dafür, dass der Prozesspfad ausgeführt wird und dass eine Marke an die nächste Elementsequenz des zweiten inklusiven Schlüssels weitergegeben wird, damit bei diesem die Prüfung für die Ausführung erfolgen kann. Dieser Ablauf wird bis zum Element $n-1$ durchgeführt. Das Element n enthält dagegen lediglich ein Assign n Element, welches wie das Assign a Element, an der aktuellen Position der Zeichenkette die auszuführende Prozessnummer ausliest. Diese wird daraufhin im Decide n Element geprüft. Wenn der inklusive Gateway keinen else-Zweig aufweist, dann ist der zweite Nachfolger des Decide Elements ein Dispose Element, um die duplizierte Marke aus dem Simulationsmodell zu entfernen. Ist jedoch ein else-Zweig vorhanden, dann ist das Decide else-Element der zweite Nachfolger des Decide n Elements. Dieses prüft die komplette Zeichenkette nach dem Wert „000“, der als eigenständiger Eintrag in der Wahrscheinlichkeitstabelle aufgeführt wird. Ist diese Prüfung wahr, dann wird der else-Zweig ausgeführt. Bei einem negativen Ergebnis wird hingegen die Marke an ein Dispose Element weitergeleitet, damit diese aus dem Modell entfernt werden kann.

Durch diese Struktur ist sichergestellt, dass nur die Marken, an das zusammenführende inklusive first come Gateway, weitergeleitet werden, die auch bei der Ausführung eines Prozesspfades verwendet wurden. Dies stellt insbesondere das einzufügende Dispose Element sicher.

Darstellung der Transformationsregel



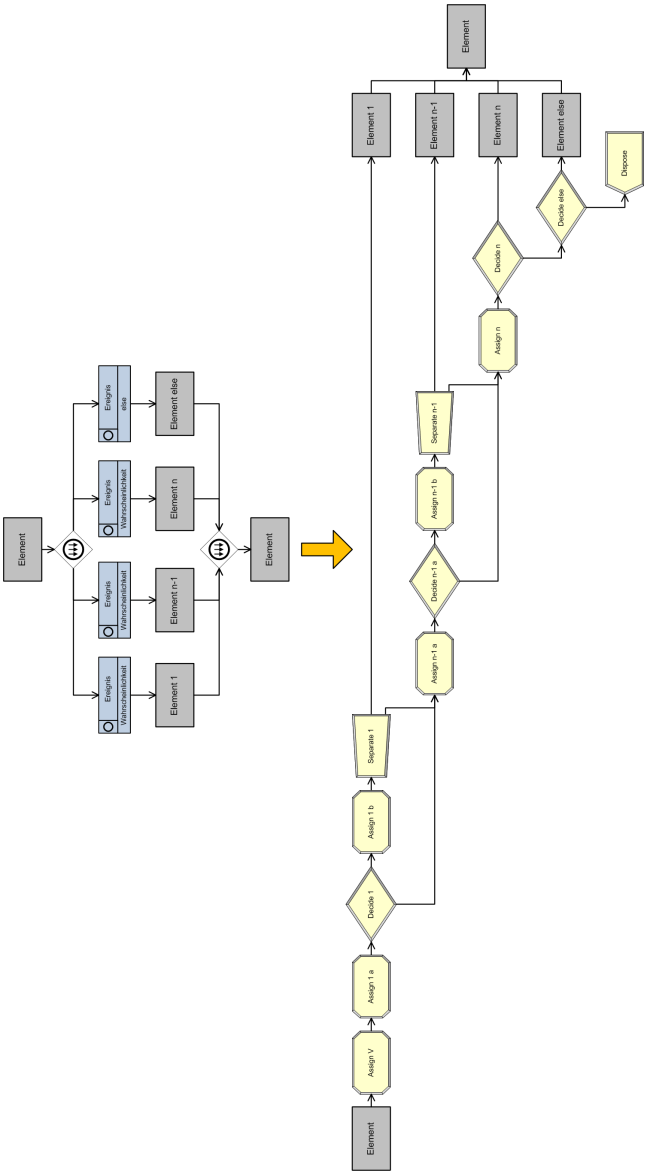
Transformationsregel $arTR_{27}$
If an every time inclusive gateway has one predecessor and more than one successor, then it will be a construct of several elements. First it will become an initial assign element. Let n+1 be the number of paths that follow the gateway. Every path until n-1 will get a sequence of an assign element, a decide element, an assign element and a separate element. The second successor of the separate element and the second successor of the decide element is the first element of the next sequence i+1. The n element is different to the previous elements and only consists of the first assign, the separate element. An else path will be a decide element that is the successor of the decide n element. The first successor of the else decide element will be the successor of the else key and the second successor will be drain.

Transformationsregel $arTR_{27}$ erzeugt das gleiche Konstrukt wie die Transformationsregel $arTR_{26}$. Dementsprechend weist die Transformationsregel die gleichen Attribute für die Elemente auf (Tabelle 34). Da auf Elementebene das gleiche Konstrukt erzeugt wird, wird auf die Ausführungen der Transformationsregel $arTR_{26}$ verwiesen.

Tabelle 34: Parameter der Elemente der Transformationsregel $arTR_{27}$

Element	Assignments/Decide
Assign V	Attribute, path, DISC(everytime(1,2), everytime(1,1), everytime(2,2), everytime(2,1), everytime(3,2), everytime(3,1), everytime(4,2), everytime(4,1), everytime(5,2), everytime(5,1), everytime(6,2), everytime(6,1), everytime(7,2), everytime(7,1), everytime(8,2), everytime(8,1)) Attribute, pos, 1
Assign 1 a	Attribute, process, Val(Mid(Str(path), pos, 1))
Assign 1 b	Attribute, pos, pos + 1
Decide 1	If Attribute process == 1
Assign n-1 a	Attribute, process, Val(Mid(Str(path), pos, 1))
Assign n-1 b	Attribute, pos, pos + 1
Decide n-1	If Attribute process == 2
Assign n	Attribute, process, Val(Mid(Str(path), pos, 1))
Decide n	If Attribute process == 3
Decide else	If Attribute path == 000

Darstellung der Transformationsregel



Transformationsregel $arTR_{28}$
If a complex gateway or a wait-for-all complex gateway has more than one predecessor and one successor, then it will be a batch element.

Transformationsregel $arTR_{28}$ ist identisch mit der Transformationsregel $arTR_{22}$. Für die Erläuterung der Regel wird entsprechend auf diese Regel verwiesen.

Transformationsregel $arTR_{29}$
If a first-come complex gateway has more than one predecessor and one successor, then it will we be a sequence of two assign elements and two decide elements. The second successor of the first decide element will be a dispose element. The first successor of the second decide element will the the second assign element and the second successor will be an additional assign element that has the the successor of the gateway as its successor.

Die Transformationsregel $arTR_{29}$ entspricht der Transformationsregel $arTR_{23}$, betrachtet jedoch das inklusive Gateway anstelle eines komplexen Gateways. Aufgrund der identischen Überführung wird auf die Transformationsregel $arTR_{23}$ für die Erläuterungen verwiesen.

Transformationsregel $arTR_{30}$
If an every-time complex gateway has more than one predecessor and one successor, then it will be removed and the predecessors of the gateway will get the successor as their successor.

Bei Transformationsregel $arTR_{30}$ wird wie bei den zwei vorhergehenden Transformationsregeln auf das Äquivalent bei den zusammenführenden inklusiven Gateways verwiesen. Dementsprechend wird für die Erläuterung auf die Transformationsregel $arTR_{24}$ verwiesen.

Transformationsregel arTR$_{31}$
If a complex gateway or a wait-for-all complex gateway has one predecessor and more than one successor, then it will be a construct of several elements. Let n+1 be the number of successors, then all successors until n-1 will be a decide element with two separate elements as its successor. Both separate elements will have the next decide element as one successor. The first separate element will have the successor of the complex key as its second successor and the the second separate element will have the combining complex gateway as its successor. The decide element of the element n will have as its first successor the successor of the complex key and as a second successor a separate element identically to the second separate element of the n-1 elements. The complex else key will be a series n of decide else elements, that check the negated values of the decide elements. All decide elements until second to last will have the next decide else element as its first successor and the combining complex gateway as its second successor. The last decide element, will have as its first successor the successor of the else key and as its second successor the combining complex gateway. If a complex key has more than one condition, then it will be more than one decide element. The next decide element will be placed between the previous decide element and the first separate element and will get as its second successor the second separate element.

Die Transformationsregel arTR$_{31}$ überführt das komplexe „wait-for-all" Gateway in mehrere Elemente in Arena. Die Transformation ist, wie beim inklusiven „wait-for-all" Gateway, in Transformationsregel arTR$_{25}$, dreigeteilt. Die Regel geht von $n+1$ Nachfolgern aus, wobei n die Anzahl der komplexen Schlüssel ist und die $+1$ der komplexe else-Schlüssel, der durch die semiautomatische Normalisierungsregel sNR$_{12}$ bei einem verzweigenden komplexen Gateway immer vorhanden ist.

Alle Prozesspfade bis zum Prozesspfad $n-1$ werden nach dem ersten Muster überführt. Dieses beinhaltet ein Decide Element sowie zwei Separate Elemente, die Nachfolger des Decide Elements sind. Das erste Separate Element hat als ersten Nachfolger den Nachfolger des entsprechenden komplexen Schlüssels und als zweiten Nachfolger das Decide Element des nächsten Prozesspfades. Das zweite Separate Element hat dieses Decide Element als ersten Nachfolger und das zusammenführende komplexe Gateway als zweiten Nachfolger.

Der Prozesspfad n hingegen besteht nur aus einem Decide Element und einem Separate Element. Ist dieser Prozesspfad auszuführen, dann muss der else-Zweig nicht mehr geprüft werden. Als ersten Nachfolger weist das Decide n Element daher den Nachfolger des komplexen Schlüssels und als zweiten Nachfolger das Separate else-Element auf. Der erste Nachfolger dieses Elements ist ein Decide else-Element und der zweite Nachfolger das zusammenführende komplexe Gateway.

Der else-Zweig soll nur ausgeführt werden, wenn kein anderer Prozesspfad ausgeführt wird. Hierfür müssen alle durchgeführten Prüfungen negiert und mit einem logischen Und verbunden werden. Da ein Decide Element jedoch nur eine Prüfung beinhalten kann, ist für jede negierte Prüfung ein separates Decide Element notwendig. Ein abstrahiertes Beispiel hierfür ist in Tabelle 35 aufgeführt. Wird ein konkretes Attribut geprüft, beispielsweise „If Attribute ordervalue >= 5000“ enthält das entsprechende Decide else-Element die Prüfung „If Attribute ordervalue < 5000“. Mittels der Negation der Bedingung wird sichergestellt, dass der else-Pfad nur ausgeführt wird, wenn kein anderes Pfad auszuführen ist.

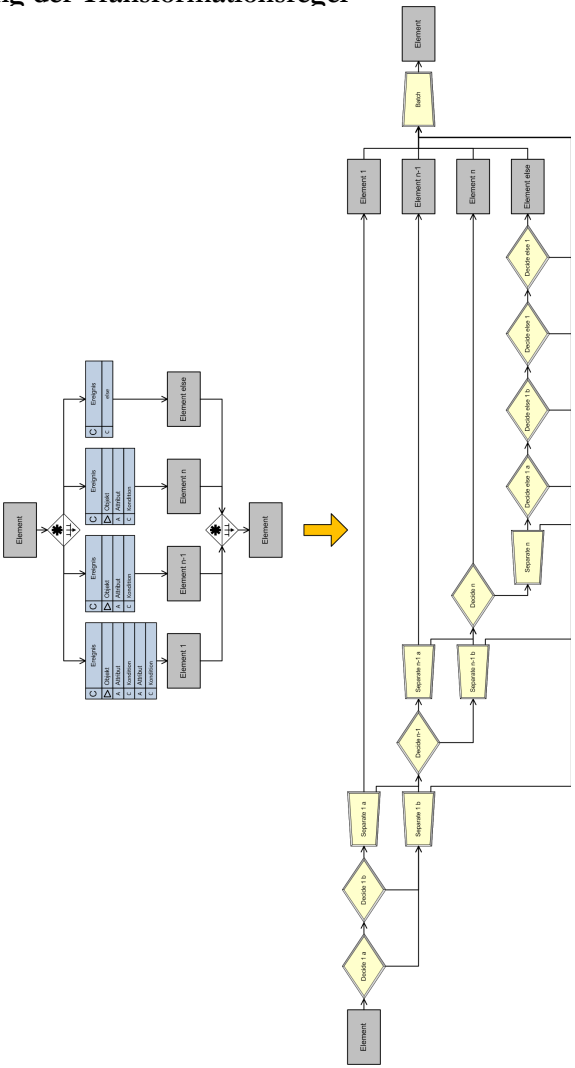
Tabelle 35: Beispiel für die logische Prüfung in der Transformationsregel arTR31

Element	Prüfung
Decide 1	$A \wedge \neg B$
Decide n-1	A
Decide n	B
Decide else 1	$\neg A \wedge B$
Decide else n-1	$\neg A$
Decide else n	$\neg B$

Die Transformationsregel $arTR_{31}$ weist jedoch noch eine Zusatzregel auf. Wenn ein komplexer Schlüssel mehr als eine Kondition beinhaltet, dann wird diese durch zusätzliche Decide Elemente berücksichtigt. Diese werden zwischen das erste Decide Element und das erste

Separate Element eingefügt und haben als zweiten Nachfolger das zweite Separate Element.

Darstellung der Transformationsregel

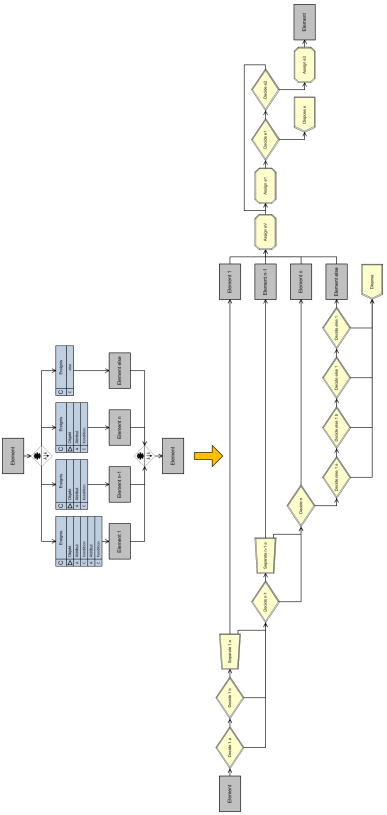


Transformationsregel $arTR_{32}$
If a first-come complex gateway has one predecessor and more than one successor, then it will be a construct of several elements. Let n+1 be the number of successors, then all successors until n-1 will be a decide element with a separate elements as its first successor. The second successor of the decide element will be the next decide element. The separate element will have the successor of the complex key as its first successor and the next decide element as its second successor. The decide element of the element n will have as its first successor the successor of the complex key and as a second successor the first decide else element. The complex else key will be a series of decide else elements, that check the negated values of the decide elements. All decide elements until the second to last will have the next decide else element as its first successor and a dispose element as its second successor. The last decide else element, will have as its first successor the successor of the else key and as its second successor the dispose element. If a complex key has more than one condition, then it will be more than one decide element. The next decide element will be placed between the previous decide element and the separate element and will get the decide element of the next complex key as its second successor.

Transformationsregel $arTR_{32}$ überführt das komplexe „first-come“ Gateway nach Arena. Hierfür ist wie bei der Transformationsregel $arTR_{31}$ mehr als ein Element nötig, ebenfalls mit der Dreiteilung. Alle Elemente bis $n-1$ erhalten ein Decide Element, in dem die Prüfung nach der Bedingung erfolgt. Im positiven Fall folgt ein Separate Element, welches den Nachfolger des komplexen Schlüssels als ersten Nachfolger aufweist und das Decide Element des nächsten komplexen Schlüssels als zweiten Nachfolger. Dieses Element ist auch der zweite Nachfolger des Decide Elements, wenn die Prüfung negativ ausfällt. Der komplexe Schlüssel n wird lediglich in ein Decide Element überführt, welches die Prüfung der Kondition beinhaltet. Ist die Kondition wahr, dann wird der Nachfolger des komplexen Schlüssels n ausgeführt, andernfalls wird die Marke an das erste Decide else-Element übergeben. Das Decide else-Element beinhalten wie in Transformationsregel $arTR_{31}$ die negativen Prüfungen aller vorangegangenen Decide Elemente. Auf die Notwendigkeit dieses Konstrukts wurde bereits bei der Transformationsregel $arTR_{31}$ eingegangen, weshalb auf diese verwiesen wird. Als Nachfolger, wenn die negierte Prüfung nicht wahr ist, haben alle Decide else Element ein Dispose

Element, womit sicher gestellt wird, dann nur die Marken an das zusammenführende komplexe Gateway weitergeleitet werden, die auch bei der Ausführung eines Prozesspfades beteiligt waren. Analog zur Transformationsregel $arTR_{31}$ hat die Transformationsregel $arTR_{32}$ die gleiche Zusatzregel. Hat ein komplexer Schlüssel mehr als eine Kondition, dann werden diese durch zusätzliche Decide Elemente im Simulationsmodell integriert. Die weiteren Decide Element werden zwischen das erste Decide Element und das erste Separate Element eingefügt und haben als zweiten Nachfolger das erste Decide Element des nachfolgenden komplexen Schlüssels.

Darstellung der Transformationsregel



Transformationsregel $arTR_{33}$
If an every time complex gateway has one predecessor and more than one successor, then it will be a construct of several elements. Let n+1 be the number of successors, then all successors until n-1 will be a decide element with a separate elements as its first successor. The second successor of the decide element will be the next decide element. The separate element will have the successor of the complex key as its first successor and the next decide element as its second successor. The decide element of the element n will have as its first successor the successor of the complex key and as a second successor the first decide else element. The complex else key will be a series of decide else elements, that check the negated values of the decide elements. All decide elements until the second to last will have the next decide else element as its first successor and a dispose element as its second successor. The last decide else element, will have as its first successor the successor of the else key and as its second successor the dispose element. If a complex key has more than one condition, then it will be more than one decide element. The next decide element will be placed between the previous decide element and the separate element and will get the decide element of the next complex key as its second successor.

Das erzeugte Modellkonstrukt der Transformationsregel $arTR_{33}$ entspricht dem der Transformationsregel $arTR_{32}$. Für die Erläuterung der Regel wird daher auf diese Transformationsregel verwiesen.

Transformationsregel $arTR_{34}$
Every resource will be added to the resource table.

Die Transformationsregel $arTR_{34}$ überführt jede Ressource in die Ressourcentabelle von Arena. Hierfür wird der Name der Ressource als „Name“ verwendet. Ist die Kapazität der Ressource fest vorgegeben, dann wird der „Type“ auf „Fixed Capacity“ eingestellt und das Attribut „Capacity“ mit der definierten Quantität belegt. Liegt hingegen ein Zeitplan vor, dann wird das Attribut „Type“ auf „Based on Schedule“ eingestellt und das Attribut „Schedule Name“ mit dem Namen des Zeitplans versehen.

Transformationsregel $arTR_{35}$
Every resource will be added to the schedule table.

Entsprechend der Transformationsregel $arTR_{34}$ überführt die Transformationsregel $arTR_{35}$ alle Zeitpläne nach Arena. Diese werden hierbei in die Schedule Tabelle überführt. Zunächst muss hierfür der Name angegeben werden. Als „Format Type“ wird im ProSiT Konzept ausschließlich der Wert „Duration“ verwendet und als „Type“ der Wert „Capacity“, damit ein Zeitplan für Ressourcen verwendet werden kann. Abhängig von den definierten Werten des Zeitplans wird das Attribut „Time Units“ entsprechend konfiguriert und in das Attribut „Durations“ die einzelnen Zeilen der Zeittabelle überführt.

Transformationsregel $arTR_{36}$
Every object will be added to the entity table.

Marken unterscheiden sich in Arena lediglich in ihrem Aussehen und den mit ihnen zusammenhängenden Kosten. Die Transformationsregel $arTR_{36}$ überführt daher jedes Objekt in die Entity-Tabelle. Als „Name“ für die Marke wird der Name des Objekts verwendet. Attribute müssen nicht definiert werden. Diese werden automatisch über Assign Elemente zugewiesen.

3.6.3 Untersuchungen zur Überführung in weitere Simulationsumgebungen

Im Rahmen der Konzeption des ProSiT Konzepts wurden neben AnyLogic und Arena noch vier weitere Simulationsumgebungen untersucht. Die Simulationsumgebung Enterprise Dynamics war Gegenstand der Diplomarbeit von Himmelreich (2007). Hierbei wurde jedoch das Problem festgestellt, wenn eine Marke eine stationäre Ressource betritt, beispielsweise einen Raum, können nur zu diesem Zeitpunkt weitere Ressourcen hinzugefügt werden. Ebenfalls ist die Freigabe der Ressourcen nur beim Austritt der Marke aus dem Raum möglich (Himmelreich 2007, S. 74).

Die Eignung der Simulationsumgebung Plant Simulation wurde im Rahmen der Bachelorarbeit von Just (2010) untersucht. Basierend auf dem Transformationsmodell aus Kloos et al. (2009) wurden Transformationsregeln zur Simulationsumgebung definiert. In der Summe wurden 16 Transformationsregeln erstellt. Besondere Betrachtung bedarf die Überführung einer Aktivität in mehrere Einzelstationen. Eine Einzelstation ist dadurch gekennzeichnet, dass zeitgleich nur eine Marke verarbeitet werden kann und hierfür Ressourcen herangezogen werden können, die als Werker definiert werden. Da nur eine Marke im gleichen Zeitraum bearbeitet werden kann, sind mehrere Einzelstationen notwendig, um eine Aktivität korrekt abzubilden (Just 2010, S. 12-13). Dies kann in Form eines separaten Netzwerks erfolgen, welches einen Eingang und einen Ausgang vorweist. Auf den Eingang folgt ein Puffer mit unbegrenzter Kapazität. Als Nachfolger hat der Puffer mehrere Einzelstationen, die jeweils eine Instanz der Aktivität abbildet (Just 2010, S. XIII-XV). Die Anzahl der notwendigen Einzelstationen für eine Aktivität kann, über die am häufigsten zur Verfügung stehende Ressource, ermittelt werden. Das Netzwerk besteht demnach aus einem Puffer und Einzelstationen. Aufgrund dieser Umsetzung konnte die Ressourcenbindung und Ressourcen-

freigabe nicht implementiert werden. Wie bei Enterprise Dynamics konnte das Zahnarzt Gedankenexperiment nicht umgesetzt werden. Der Grund liegt bei Plant Simulation am Werkerkonzept, welches für die Ressourcen verwendet wurde. Wird dieses nicht verwendet und statt dessen ein eigenes Ressourcenmodell entwickelt, dann ist eine Überführung des ProSiT Ablaufdiagramms nach Plant Simulation erneut zu prüfen.

Die Problematik der Umsetzung des Zahnarzt-Gedankenexperiments ergab sich auch bei der dritten untersuchten Simul8. So konnte die Reservierung der Ressourcen nicht realisiert werden, sodass ein sicherer Ablauf des Prozessmodells nicht gewährleistet ist. Problematisch stellte sich auch die Umsetzung von inklusiven Entscheidungen dar. Während das wait-for-all und every-time Konzept umgesetzt werden konnte, war dies beim first-come Fall nicht möglich. Erste Untersuchungen zeigen aber, dass eine Überführung möglich ist, wahrscheinlich aber mehrere Anwendbarkeitsregeln definiert werden müssen, welche nicht unterstützte Konzepte ausschließt.

Erste Untersuchungen wurden auch zum Simulationspaket DESMO-J (Meyer et al. 2005, S. 263) durchgeführt. Aufgrund der Freiheit unterschiedliches Verhalten direkt in Java zu implementieren ergaben erste Tests, dass das ProSiT Ablaufdiagramm in diesem Simulationspaket abgebildet werden kann. Da eine grafische Repräsentation bis zur Version 2.3.01 jedoch nicht unterstützt wurde, stellt dieses Simulationspaket ebenfalls einen offenen Forschungspunkt dar.

Generell ergibt sich in Bezug auf weitere Simulationsumgebungen die Forschungsmöglichkeit allgemeine Kriterien und Experimente zu definieren, die verwendet werden können, um die Eignung einer Simulationsumgebung zu untersuchen. Hierfür sollte für jedes Element des ProSiT Ablaufdiagramms ein Experiment definiert werden. Kann in einer Simulationsumgebung dieses nicht umgesetzt werden, dann ist dieses Element im Rahmen von Anwendbarkeitsregeln zu berücksichtigen und entsprechend auszuschließen.

4 Evaluation des Transformationsmodell Ansatzes

4.1 Konzept der Evaluation

Nach dem Memorandum der gestaltungsorientierten Wirtschaftsinformatik (Österle et al. 2010a, S. 668) sind Begutachtungsverfahren für wissenschaftliche Publikationen ein Teil der Evaluation von Ergebnissen und Erkenntnissen der gestaltungsorientierten Wirtschaftsinformatik. In diesem Sinne wird als Teil der Evaluation auf die Arbeiten Kloos et al. (2009), Kloos et al. (2010) und Kloos et al. (2011) verwiesen. Gegenstand dieser Artikel war jeweils das konzipierte Transformationsmodell. Nicht evaluiert wurde in diesen Artikeln die Regelbasis. Eine strenge Falsifikation im Sinne von Popper kann jedoch aus Platzgründen im Rahmen dieser Arbeit nicht erfolgen. Statt dessen soll die Evaluation der Regelbasis anhand von jeweils zwei beispielhaften Modellen erfolgen. Dies ist dem Platzbedarf geschuldet, den die Erläuterung der Evaluation anhand der einzelnen Modellen erfordert. Eine Übersicht über die für die Evaluierung verwendeten Modelle ist in Tabelle 36 aufgeführt.

Das Modell der Produkteinführung stammt aus dem Artikel von Overhage et al. (2011). In diesem ist das Modell als eEPK (Overhage et al. 2011, S. 750) sowie als UML Aktivitätsdiagramm (Overhage et al. 2011, S. 749) abgebildet. Aus beiden Modellen wurde ein äquivalentes Modell in die Notation der Business Process Model and Notation abgeleitet. Mit diesem Modell soll geprüft werden, ob die Transformationsregeln für die einzelnen Prozessmodellierungsnotationen jeweils das gleiche Transformationsmodell erzeugen. Bei der Betrachtung der beiden Modelle in Overhage et al. (2011) und den formulierten Transformationsregeln ist jedoch nicht mit einem identischen Modell zu rechnen. Ein Beispiel hierfür ist das Zeitereignis im Aktivitätsdiagramm, welches in dieser Form bei der eEPK als

Ereignis modelliert wurde. Zu beachten ist, dass das Geschäftsprozessmodell der Produkteinführung nicht als ein simulationswürdiges Modell aufzufassen ist, sondern lediglich der Evaluation der Transformationsregeln zum ProSiT Ablaufdiagramm dient.

Tabelle 36: Verwendete Modelle zur Evaluierung

Regelwerk	Modell
eEPK als Quellmodell	Produkteinführung (eEPK)
	Chirurgischer Notfall
	Voruntersuchung
BPMN als Quellmodell	Produkteinführung (BPMN)
	Einkaufsprozess
UML Aktivitätsdiagramm als Quellmodell	Produkteinführung (UML)
	Kfz vermieten
Anwendung der Normalisierung	Chirurgischer Notfall
	Voruntersuchung
AnyLogic als Zielumgebung	Chirurgischer Notfall (normalisiert)
	Voruntersuchung (normalisiert)
Arena als Zielumgebung	Chirurgischer Notfall (normalisiert)
	Voruntersuchung (normalisiert)

Mit den zwei Modellen „Chirurgischer Notfall“ und „Voruntersuchung“ liegen jedoch zwei Modelle vor, die jeweils im Rahmen einer Prozessmodellierung erhoben und im Rahmen eines Simulationsvorhabens verwendet wurden. Da beide Modelle aber als eEPK modelliert wurden, diese jedoch auch für die Normalisierung sowie die Transformation in die Simulationsumgebungen verwendet werden können, werden die Transformationsregeln auf drei eEPK-Modelle angewendet. Da beide Geschäftsprozessmodelle basierend auf einem Unternehmen erstellt wurden, entsprechen diese der von Riege et al. (2009, S. 81) geforderten Evaluation gegen die Realwelt.

Als zweites Modell für BPMN wird das Geschäftsprozessmodell eines „Einkaufsprozesses“ verwendet. Dieses Modell stellt eine Lösung für die Aufgabe des Business Process Reengineering dar, die im Rahmen der Übung Geschäftsprozessmanagement an der Technischen Universität Ilmenau gestellt wird (Kloos 2010, S. 91). Bei diesem Modell sind möglichst viele Elemente der BPMN untergebracht. Das Modell stellt daher auch kein simulationswürdiges Modell dar.

Für das UML-Aktivitätsdiagramm wird als zweites Modell der Geschäftsprozess „Kfz vermieten“ aus Oestereich et al. (2004, S. 112) verwendet. In ihrem Buch beschreiben die Autoren, wie mittels der UML eine objektorientierte Geschäftsprozessmodellierung realisiert werden kann, weshalb aus diesem Buch ein Prozess entnommen wird, um die entsprechenden Transformationsregeln zu überprüfen.

Bei der Auswahl der Geschäftsprozessmodelle für die Evaluation wurde darauf geachtet, dass Modelle von anderen Autoren verwendet oder Modelle, die nicht explizit für die Evaluation der Regelbasis modelliert wurden. Lediglich das Geschäftsprozessmodell der Produkteinführung in der Notation BPMN entspricht nicht diesen Grundzügen. Wie aber bereits aufgeführt, dient dieses Modell zum Vergleich der erzeugten konzeptionellen Transformationsmodelle und konnte nur durch eine eigene Modellierung explizit für die Evaluation als Grundlage verwendet werden.

Der Aufbau der Evaluation der einzelnen Regelbasen wird nachfolgend kurz erläutert. Für die Evaluation der Transformationsregeln zur Überführung der Geschäftsprozessmodelle in das ProSiT Ablaufdiagramm wird zunächst das Ausgangsmodell dargestellt. Anschließend werden Vorbereitungsregeln auf dieses Modell angewendet. Bei der eEPK werden hierfür die Regeln, in einer Tabelle aufgeführt, bei BPMN hingegen erfolgt eine grafische Darstellung und beim UML Aktivitätsdiagramm werden die Vorbereitungsregeln sowohl in Tabellenform, als auch grafisch dargestellt. Die reduzierte Darstellung in Tabellenform wird verwendet, wenn lediglich Elemente aus dem

Modell entfernt werden. Die Anwendung der eigentlichen Transformationsregeln erfolgt jeweils grafisch bei allen Notationen. Lediglich die Regel, welche eine Aktivität im ProSiT Ablaufdiagramm erzeugt, wird nicht explizit aufgeführt, da die Anwendung der Regel jeweils identisch ist und sich direkt auf ein Element bezieht und nicht dessen Kontext berücksichtigen muss. Die grafische Darstellung der einzelnen Regeln folgt dem Gedanken der Nachvollziehbarkeit der Evaluation. Durch die ausdrückliche Offenlegung der Anwendung der einzelnen Regeln kann auf einfache Art jede angewendete Regel überprüft werden.

Die Anwendung der Transformationsregeln kann nicht als eine eigenständige Evaluation aufgefasst werden, lediglich als Anwendung der Normalisierungsregeln. Da beide betrachteten Modelle der gleichen Domäne zuzuordnen sind, wird zunächst ein Repository für die Krankenhaus Domäne definiert, das für die Anwendung einiger Transformationsregeln benötigt wird. Die anschließende Darlegung der Normalisierungsregeln ist zweigeteilt. Zunächst werden erste Normalisierungsregeln auf das, durch das Quellmodell erzeugte, konzeptionelle ProSiT Ablaufdiagramm angewendet. Anschließend erfolgt die grafische Darstellung mit dem Zwischenstand, der durch die Anwendung der Normalisierungsregeln erreicht wurde. Daraufhin werden auf diesen Zwischenstand jeweils weitere Normalisierungsregeln angewendet und abschließend das konsistente Transformationsmodell dargestellt.

Das konsistente Transformationsmodell wird anschließend als Grundlage für die Evaluation der Transformationsregeln zur Erzeugung eines Simulationsmodells verwendet. Hierfür werden die Regeln gemäß ihrer Prämissen angewendet und jeweils grafisch dargestellt, wie dies bei der Überführung der Quellmodelle ebenfalls der Fall ist. Ebenfalls wird die Überführung der Aktivitäten nicht explizit dargestellt, da sich die Regeln nur auf ein Element beziehen. Als Endergebnis steht jeweils ein simulierfähiges Simulationsmodell.

4.2 Evaluation der Überführung der Quellmodelle in das Transformationsmodell

4.2.1 eEPK als Quellmodell

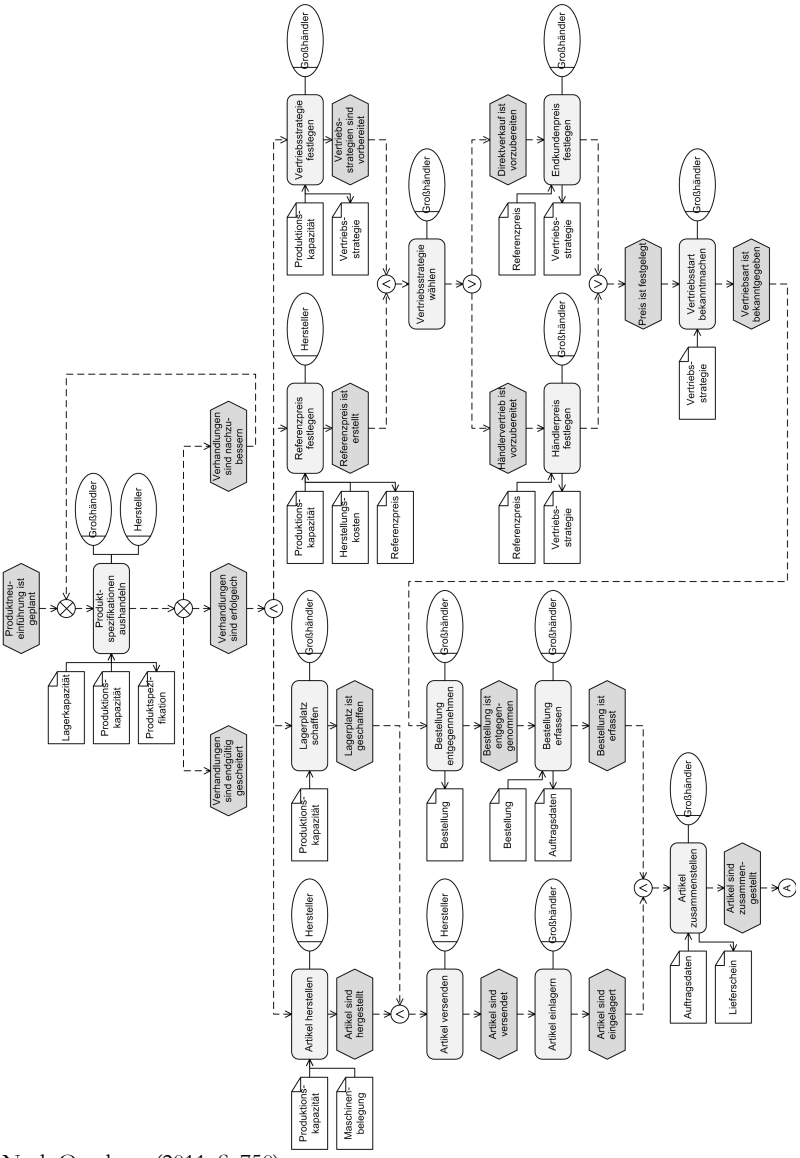
Bei der Evaluation der Transformationsregeln von der eEPK zum ProSiT Ablaufdiagramm wird zunächst der Prozess der Produkteinführung untersucht. Anschließend folgen die Teilprozesse des Chirurgischen Notfalls und der Voruntersuchung.

Produkteinführung

In den Abbildungen 46 und 47 ist die eEPK der Produkteinführung aufgeführt. Dieser wurde aus Gründen der Übersichtlichkeit in zwei Teile unterteilt. Das mit A bezeichnete Element gehört daher nicht zum Modell und verdeutlicht lediglich, an welcher Stelle der Prozess fortgesetzt wird.

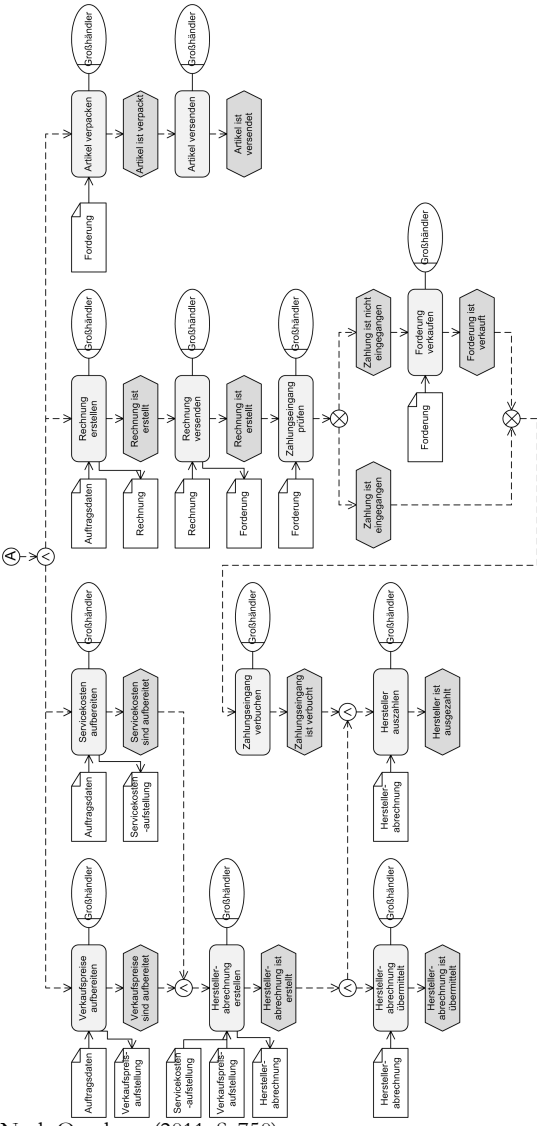
Die eEPK der Produkteinführung entspricht den von Mendling und Nüttgens (2003a, S. 21-24) aufgeführten formalen Spezifikationen. Weiterführende Syntaxregeln sind für die eEPK nicht definiert.

Nachfolgend werden ausschließlich Vorbereitungsregeln und Transformationsregeln auf dieses Geschäftsprozessmodell angewendet.



Nach Overhage (2011, S. 750)

Abbildung 46: eEPK – Produkteinführung – Teil 1



Nach Overhage (2011, S. 750)

Abbildung 47: eEPK – Produkteinführung – Teil 1

Anwendung der Vorbereitungsregeln

Die angewendeten Vorbereitungsregeln auf die eEPK werden in Tabellenform dargestellt. Mit Ausnahme der Vorbereitungsregel ePR_5 werden lediglich Zwischenereignisse entfernt. Im Produkteinführungsprozess wird diese Regel nicht angewendet.

Tabelle 37: Anwendung der eEPK Vorbereitungsregeln auf die Produkteinführung

Ereignis	Vorbereitungsregel	Aktion
14 Tage sind vergangen	ePR_1	Ereignis entfernt
Artikel ist verpackt	ePR_1	Ereignis entfernt
Artikel sind eingelagert	ePR_1	Ereignis entfernt
Artikel sind hergestellt	ePR_1	Ereignis entfernt
Artikel sind versendet	ePR_1	Ereignis entfernt
Artikel sind zusammengestellt	ePR_1	Ereignis entfernt
Bestellung ist entgegengenommen	ePR_1	Ereignis entfernt
Bestellung ist erfasst	ePR_1	Ereignis entfernt
Forderung ist verkauft	ePR_1	Ereignis entfernt
Herstellerabrechnung ist erstellt	ePR_1	Ereignis entfernt
Lagerplatz ist geschaffen	ePR_1	Ereignis entfernt
Preis ist festgelegt	ePR_3	Ereignis entfernt
Vertriebsart ist bekanntgegeben	ePR_1	Ereignis entfernt
Rechnung ist erstellt	ePR_1	Ereignis entfernt
Referenzpreis ist erstellt	ePR_1	Ereignis entfernt
Servicekosten sind aufbereitet	ePR_1	Ereignis entfernt
Verkaufspreise sind aufbereitet	ePR_1	Ereignis entfernt
Vertriebsstrategien sind vorbereitet	ePR_1	Ereignis entfernt
Zahlungseingang ist verbucht	ePR_1	Ereignis entfernt

Aus der Anwendung der Vorbereitungsregeln in Tabelle 37 resultiert die vorbereitete eEPK in den Abbildungen 48 und 49. Mit Ausnahme des

Ereignisses „Preis ist festgelegt“ wurde ausschließlich die Vorbereitungsregel ePR_1 angewendet.

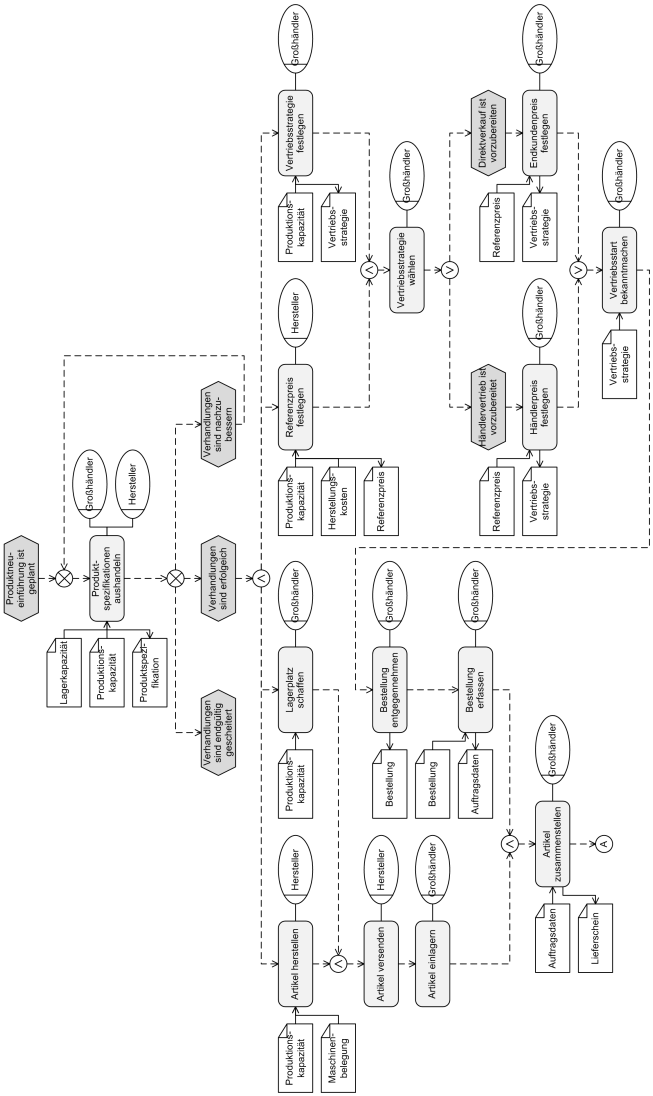


Abbildung 48: eEPK – Produkteinführung – vorbereitet – Teil 1

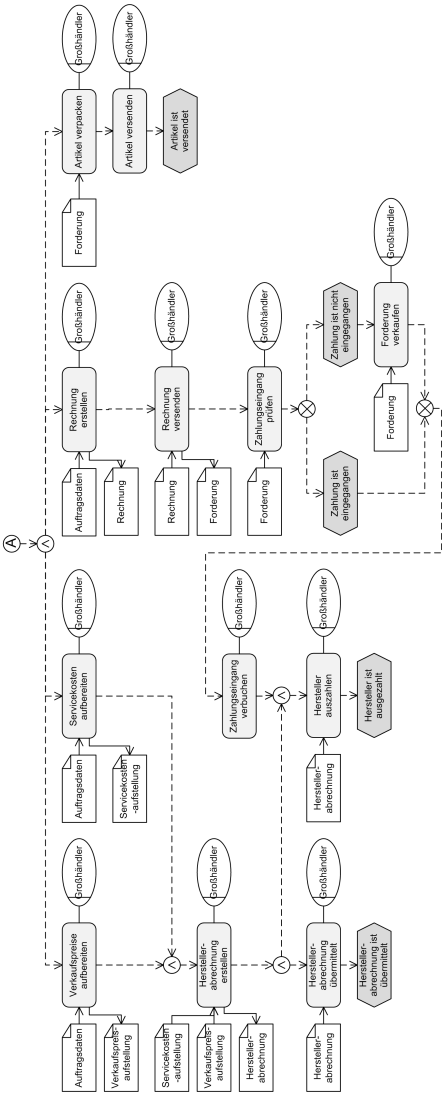
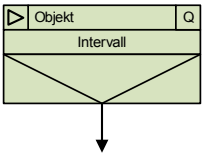
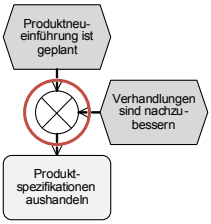
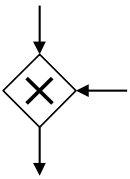


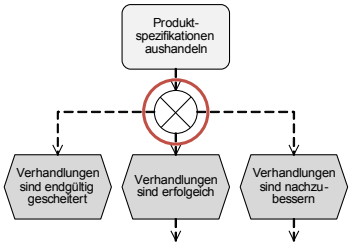
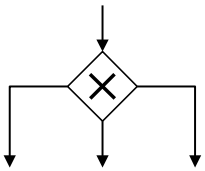
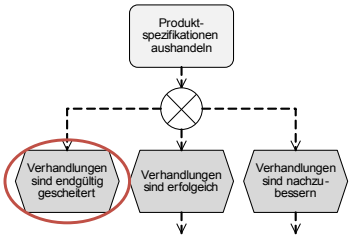
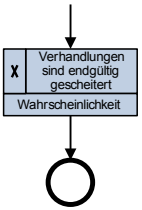
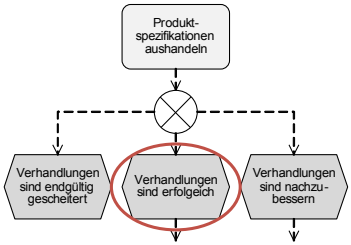
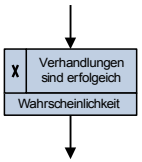
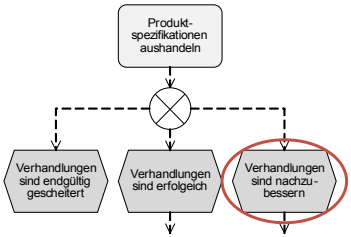
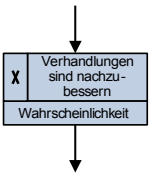
Abbildung 49: eEPK – Produkteinführung – vorbereitet – Teil 2

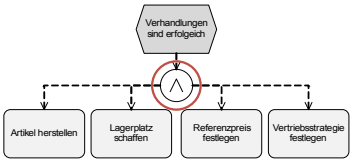
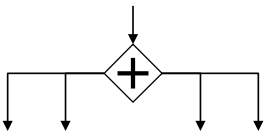
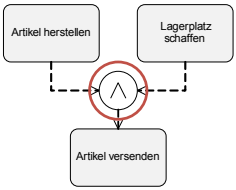
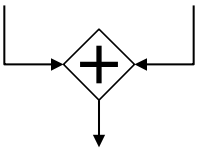
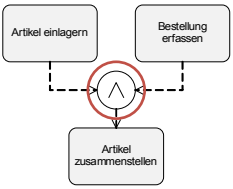
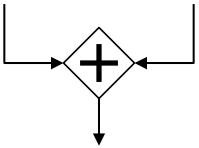
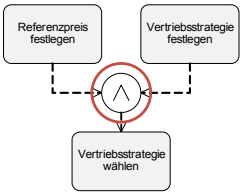
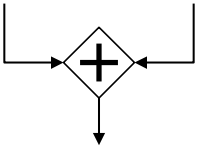
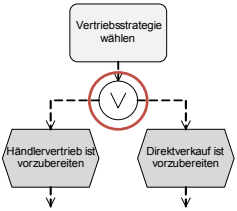
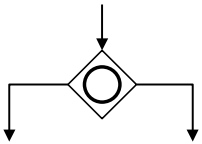
Anwendung der Transformationsregeln

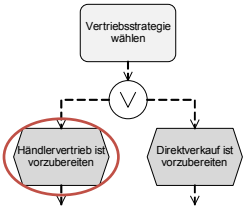
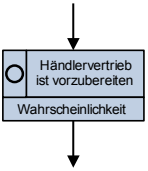
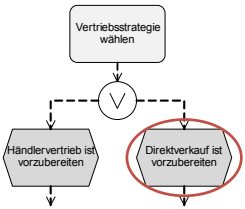
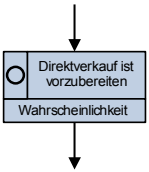
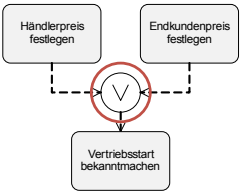
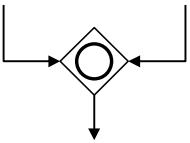
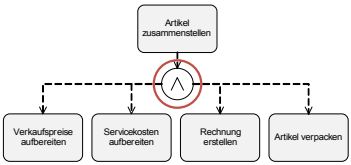
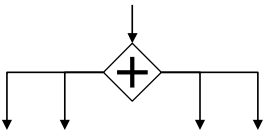
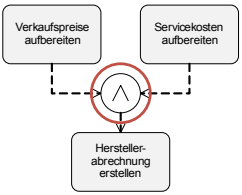
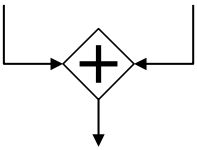
Aufbauend auf dem vorbereiteten Geschäftsprozessmodell können die Transformationsregeln angewendet werden. Die Anwendung der Transformationsregeln wird ausführlicher dargestellt, als die Vorbereitungsregel. Eine Ausnahme davon bilden die Funktionen. Da keine Hinterlegungen vorhanden sind, findet die Transformationsregel eTR_{13} keine Anwendung. Somit wird auf jede Funktion die Transformationsregel eTR_{12} angewendet. Die Zusatzregel von eTR_{12} wird nicht ausgelöst, da Organisationseinheiten an die Funktionen angeheftet sind, die Zusatzregel jedoch Stellen fordert. Eine separate Darstellung der Transformation der einzelnen Funktionen erfolgt nicht. Jede Funktion wird in eine Aktivität überführt und die Bezeichnung der Funktion wird als Tätigkeit der Aktivität verwendet. Die Anwendung der Transformationsregeln auf die restlichen Elemente ist in Tabelle 38 dargestellt.

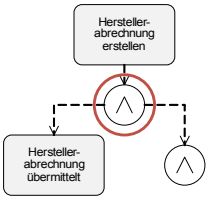
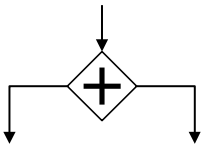
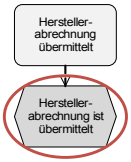
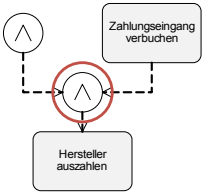
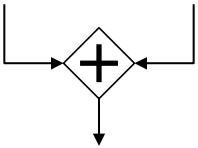
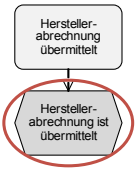
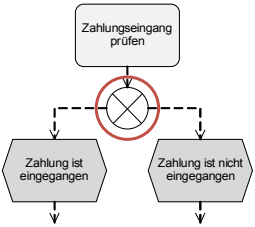
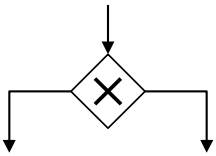
Tabelle 38: Anwendung der eEPK Transformationsregeln auf die Produkteinführung

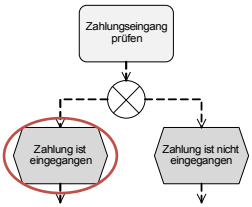
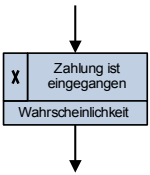
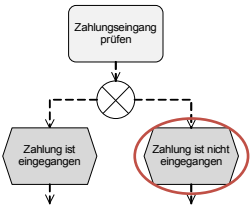
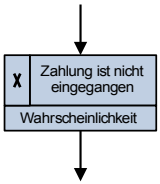
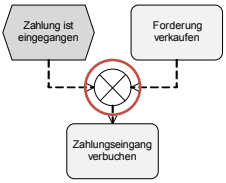
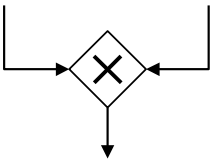
eEPK Element	eTR	Resultierendes ProSiT Ablaufdiagramm
	eTR_1	
	eTR_{19}	

eEPK Element	eTR	Resultierendes ProSiT Ablaufdiagramm
	eTR_{18}	
	eTR_{7}	
	eTR_{9}	
	eTR_{9}	

eEPK Element	eTR	Resultierendes ProSiT Ablaufdiagramm
	$e'TR_{14}$	
	$e'TR_{15}$	
	$e'TR_{15}$	
	$e'TR_{16}$	
	$e'TR_{16}$	

eEPK Element	eTR	Resultierendes ProSiT Ablaufdiagramm
	eTR_8	
	eTR_8	
	eTR_{17}	
	eTR_{14}	
	eTR_{15}	

eEPK Element	eTR	Resultierendes ProSiT Ablaufdiagramm
	eTR_{14}	
	eTR_{2}	
	eTR_{15}	
	eTR_{2}	
	eTR_{18}	

eEPK Element	eTR	Resultierendes ProSiT Ablaufdiagramm
	eTR_9	
	eTR_9	
	eTR_{19}	
	eTR_2	

Konzeptuelles Transformationsmodell

Durch die angewendeten Transformationsregeln in Tabelle 38 und der Transformationsregel eTR_{12} für die Funktionen resultiert das in den Abbildungen 50 und 51 aufgeführte konzeptuelle Transformationsmodell.

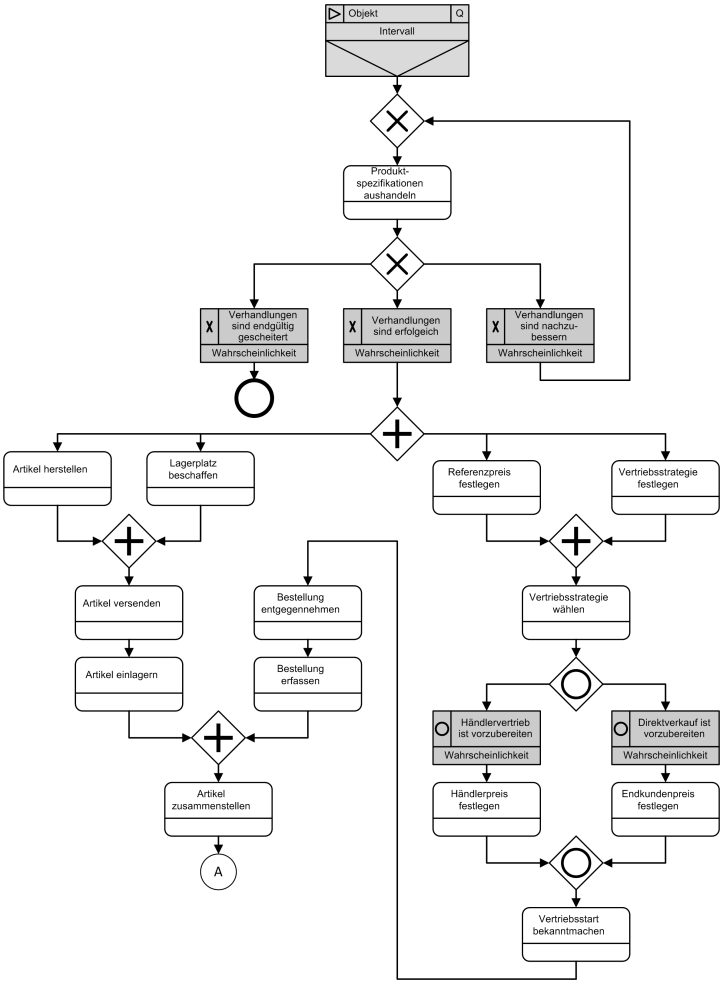


Abbildung 50: Transformationsmodell – Produkteinführung (eEPK) – konzeptuell – Teil 1

Wird das resultierende konzeptuelle Transformationsmodell hinsichtlich syntaktischer Korrektheit analysiert, dann muss geschlussfolgert werden: Das Modell ist syntaktisch korrekt. Quelle und Senke haben jeweils einen ausgehenden, beziehungsweise einen eingehenden Prozesspfad; Gateways entweder einen Vorgänger und mehrere

Nachfolger oder mehrere Vorgänger und einen Nachfolger und die Aktivität weist einen Vorgänger und einen Nachfolger auf. Auf verzweigende inklusive und exklusive Gateways folgen entsprechende Schlüssel, womit die Wahrscheinlichkeit für das Ausführen eines Prozesspfades hinterlegt werden kann. Wenn die Abbildungen 48 und 49 mit den Abbildungen 50 und 51 direkt verglichen, so kann geschlussfolgert werden, dass beide den gleichen Ablauf aus semantischer Sicht beschreiben.

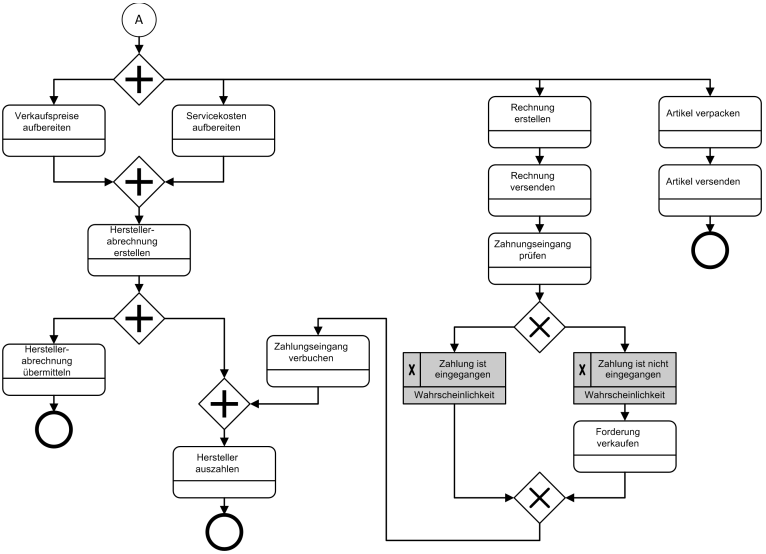


Abbildung 51: Transformationsmodell – Produkteinführung (eEPK) – konzeptuell – Teil 2

Chirurgischer Notfall

Das zweite Geschäftsprozessmodell, welches zur Evaluation der Transformationsregeln der eEPK herangezogen wird, beschreibt einen chirurgischen Notfall in einer Notfallaufnahme. Das Modell wurde im Rahmen einer Prozessanalyse aufgenommen und entsprechend modelliert. In Himmelreich (2007) ist es eines der betrachteten Modelle. Der Ablauf des Geschäftsprozessmodells ist in den Abbildungen 52

und 53 aufgeführt. Das dargestellte Modell ist jedoch nicht das ursprüngliche Modell. So wurden kleinere Fehler korrigiert und der Ablauf angepasst.

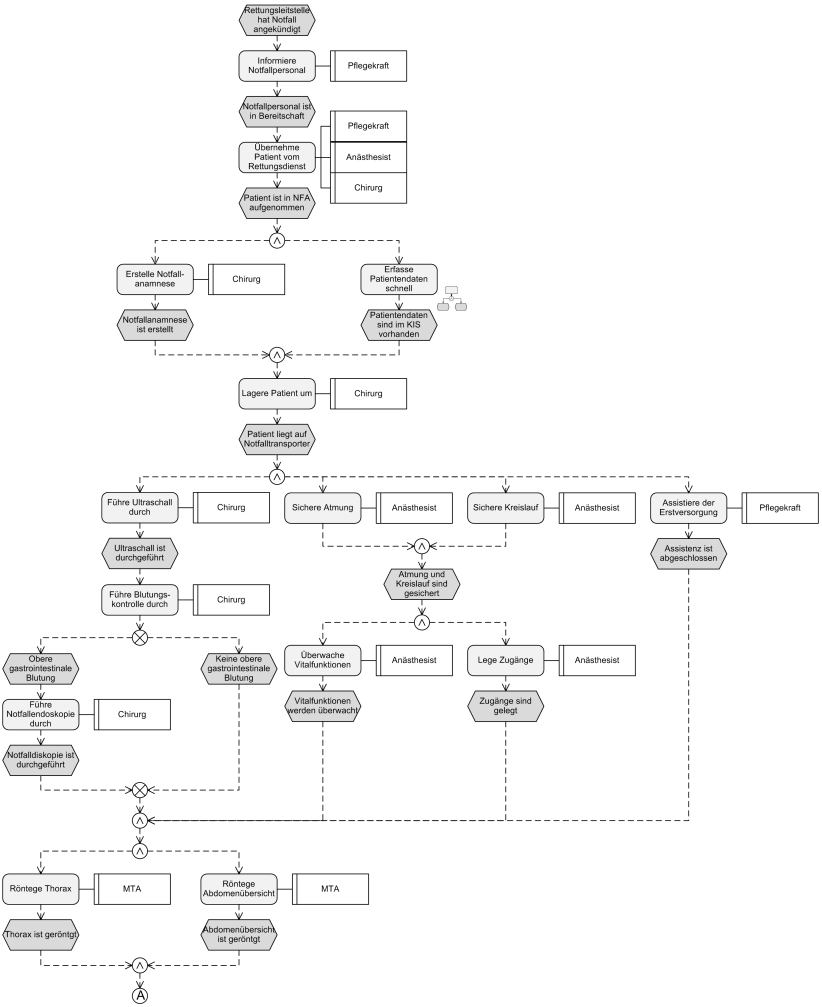


Abbildung 52: eEPK – Chirurgischer Notfall – Teil 1

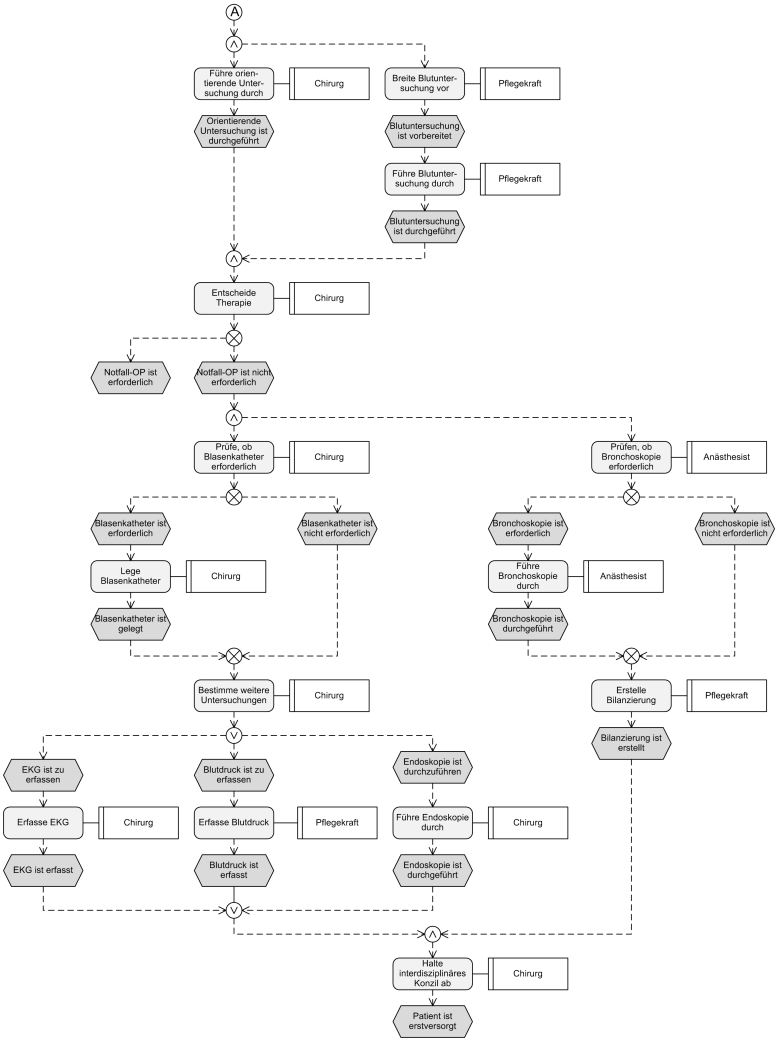


Abbildung 53: eEPK – Chirurgischer Notfall – Teil 2

Anwendung der Vorbereitungsregeln

Die Vorbereitungsregeln ePR werden auf das dargestellte Geschäftsprozessmodell angewendet. Die korrespondierenden Transformationsregeln sind für die betroffenen Ereignisse in Tabelle 39 aufgeführt. Hierbei kommt mit einer Ausnahme lediglich die Vorbereitungsregel ePR_1 zum Einsatz.

Tabelle 39: Anwendung der eEPK Vorbereitungsregeln auf den chirurgischen Notfall

Ereignis	Vorbereitungsregel	Aktion
Notfallpersonal ist in Bereitschaft	ePR_1	Ereignis entfernt
Patient ist in NFA aufgenommen	ePR_1	Ereignis entfernt
Notfallanamnese ist erstellt	ePR_1	Ereignis entfernt
Patientendaten sind im KIS vorhanden	ePR_1	Ereignis entfernt
Patient liegt auf Notfalltransporter	ePR_1	Ereignis entfernt
Ultraschall ist durchgeführt	ePR_1	Ereignis entfernt
Notfalldiskopie ist durchgeführt	ePR_1	Ereignis entfernt
Atmung und Kreislauf sind gesichert	ePR_2	Ereignis entfernt
Vitalfunktionen werden überwacht	ePR_1	Ereignis entfernt
Zugänge sind gelegt	ePR_1	Ereignis entfernt
Assistenz ist abgeschlossen	ePR_1	Ereignis entfernt
Thorax ist geröntgt	ePR_1	Ereignis entfernt
Abdomenübersicht ist geröntgt	ePR_1	Ereignis entfernt
Orientierende Untersuchung ist durchgeführt	ePR_1	Ereignis entfernt
Blutuntersuchung ist vorbereitet	ePR_1	Ereignis entfernt
Blutuntersuchung ist durchgeführt	ePR_1	Ereignis entfernt
Blasenkatheter ist gelegt	ePR_1	Ereignis entfernt
EKG ist erfasst	ePR_1	Ereignis entfernt
Blutdruck ist erfasst	ePR_1	Ereignis entfernt

Ereignis	Vorbereitungsregel	Aktion
Endoskopie ist durchgeführt	ePR_1	Ereignis entfernt
Bronchoskopie ist durchgeführt	ePR_1	Ereignis entfernt
Bilanzierung ist erstellt	ePR_1	Ereignis entfernt

Nach der Anwendung der Vorbereitungsregeln auf das Geschäftsprozessmodell resultiert das vorbereitete Modell, welches in den Abbildungen 54 und 55 aufgeführt ist.

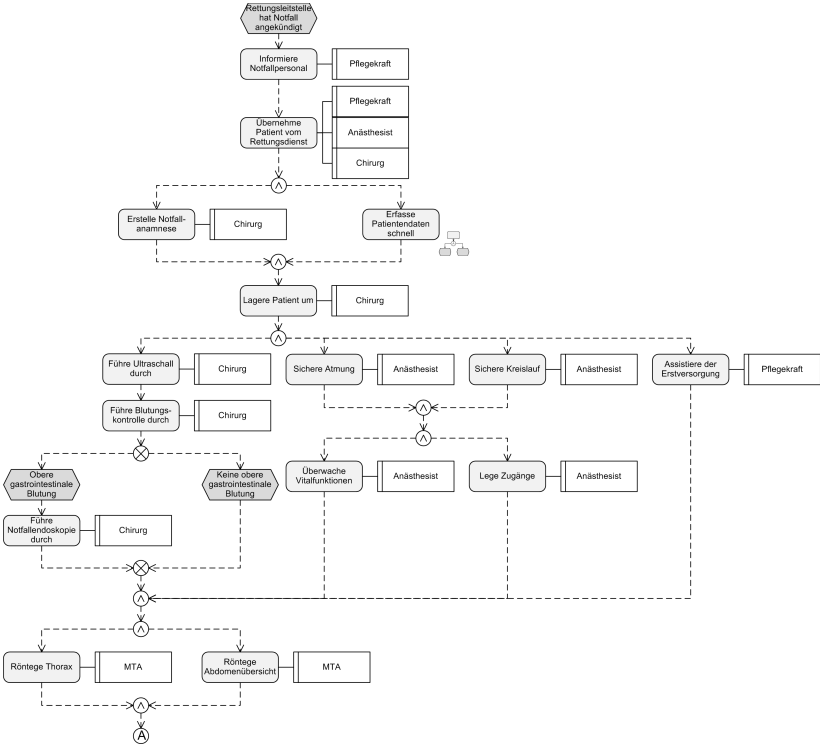


Abbildung 54: eEPK – Chirurgischer Notfall – vorbereitet – Teil 1

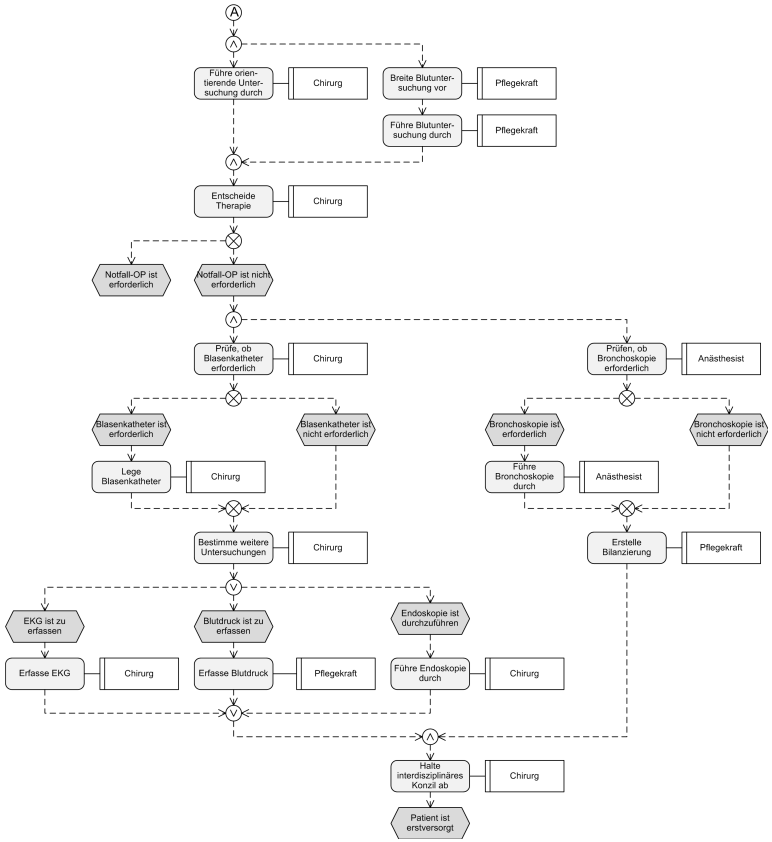


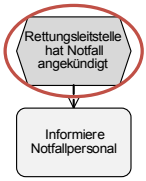
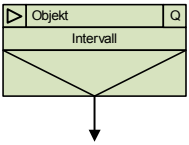
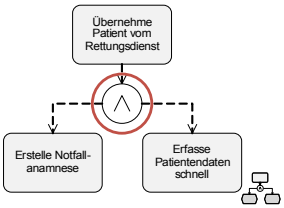
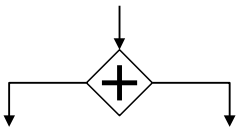
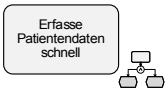
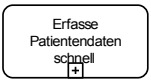
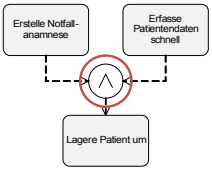
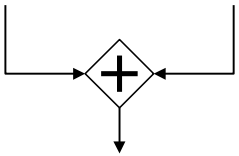
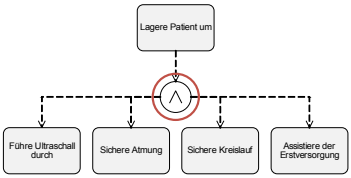
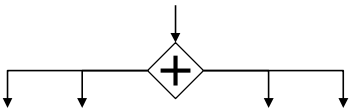
Abbildung 55: eEPK – Chirurgischer Notfall – vorbereitet – Teil 2

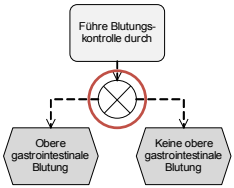
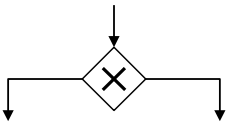
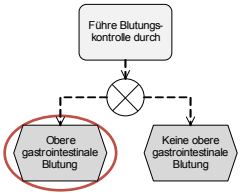
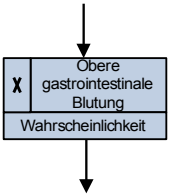
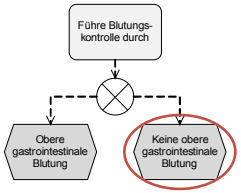
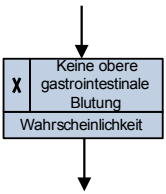
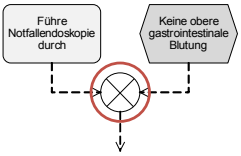
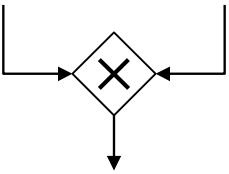
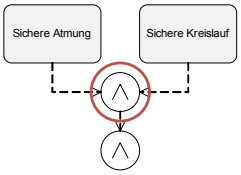
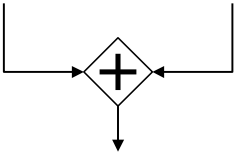
Anwendung der Transformationsregeln

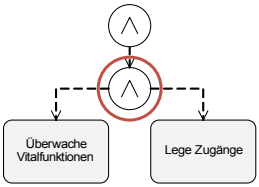
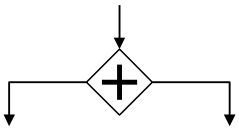
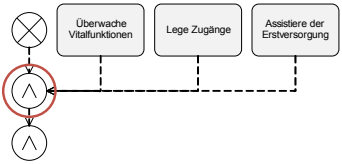
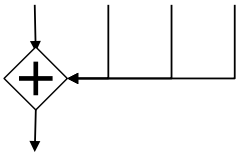
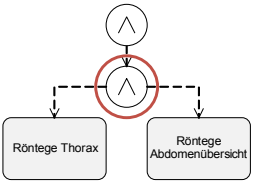
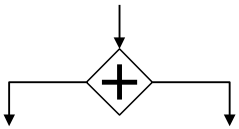
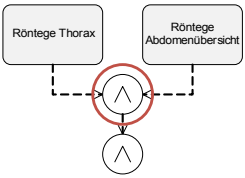
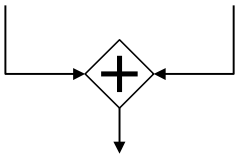
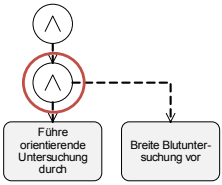
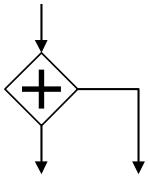
Wie beim ersten Geschäftsprozessmodell der Produkteinführung erfolgt die Anwendung der Transformationsregeln auf das vorbereitete Modell. Im Gegensatz zum vorherigen Geschäftsprozessmodell werden beim chirurgischen Notfall Stellen verwendet. Entsprechend wird bei Transformation eTR_{12} die Zusatzbedingung ausgelöst und die Stellen als Ressourcen der resultierenden Aktivität hinzugefügt. In Tabelle 40 sind, mit Ausnahme der Funktionen, die mit der Transformationsregel

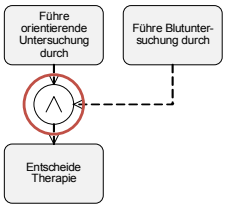
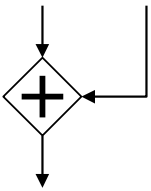
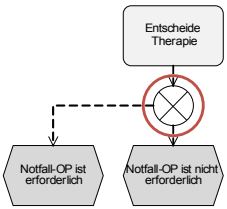
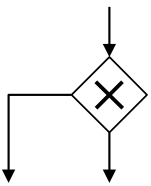
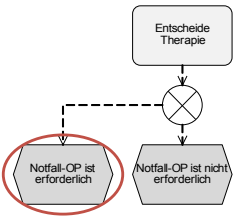
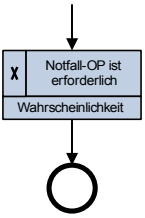
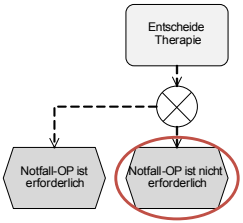
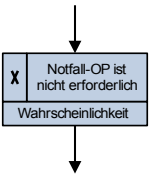
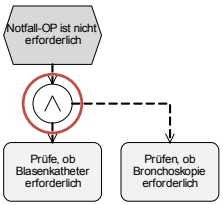
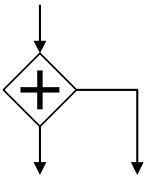
eTR_{12} überführt werden, die angewendeten Transformationsregeln auf die Elemente des vorbereiteten Modells aufgeführt.

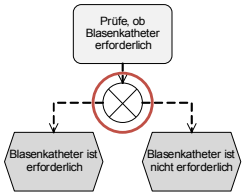
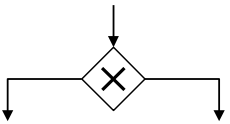
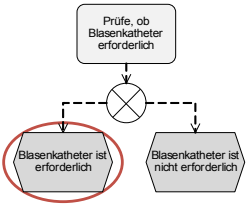
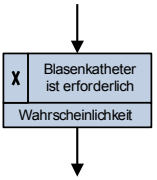
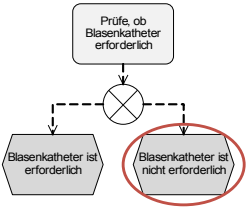
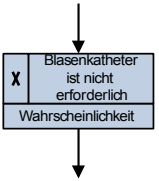
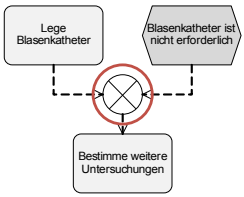
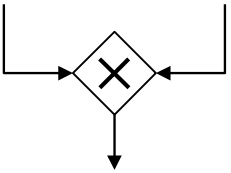
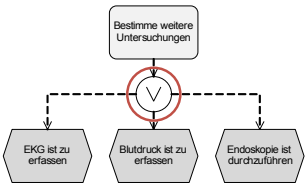
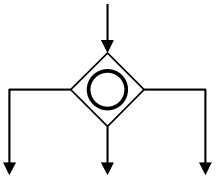
Tabelle 40: Anwendung der eEPK Transformationsregeln auf den chirurgischen Notfall

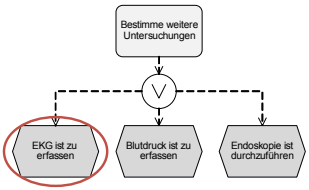
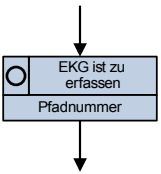
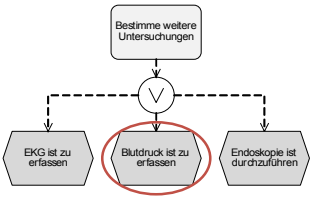
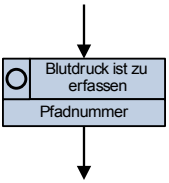
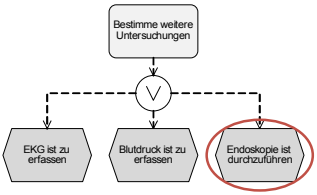
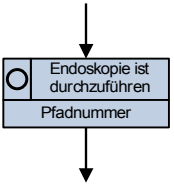
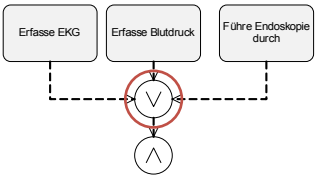
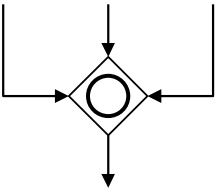
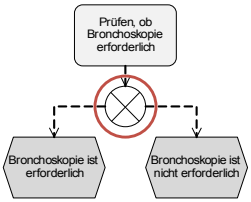
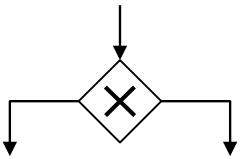
eEPK Element	eTR	Resultierendes ProSiT Ablaufdiagramm
	eTR_1	
	eTR_{14}	
	eTR_{13}	
	eTR_{15}	
	eTR_{14}	

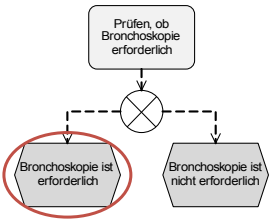
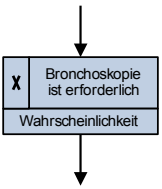
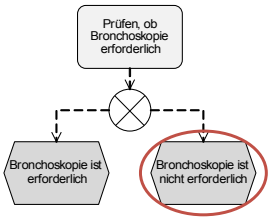
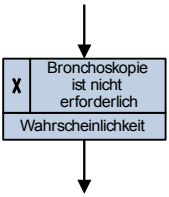
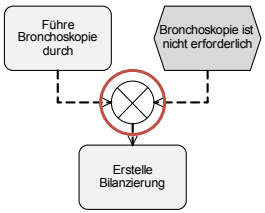
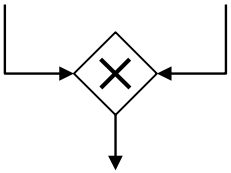
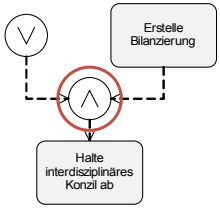
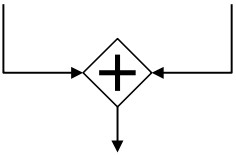
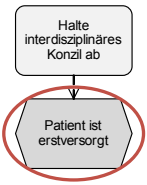
eEPK Element	eTR	Resultierendes ProSiT Ablaufdiagramm
	eTR1$_8$	
	eTR$_9$	
	eTR$_9$	
	eTR$_{19}$	
	eTR$_{15}$	

eEPK Element	eTR	Resultierendes ProSiT Ablaufdiagramm
	eTR_{14}	
	eTR_{15}	
	eTR_{14}	
	eTR_{15}	
	eTR_{14}	

eEPK Element	eTR	Resultierendes ProSiT Ablaufdiagramm
	eTR$_{15}$	
	eTR$_{18}$	
	eTR$_{7}$	
	eTR$_{9}$	
	eTR$_{14}$	

eEPK Element	eTR	Resultierendes ProSiT Ablaufdiagramm
	eTR_{18}	
	eTR_{9}	
	eTR_{9}	
	eTR_{19}	
	eTR_{16}	

eEPK Element	eTR	Resultierendes ProSiT Ablaufdiagramm
	eTR_8	
	eTR_8	
	eTR_9	
	eTR_{17}	
	eTR_{18}	

eEPK Element	eTR	Resultierendes ProSiT Ablaufdiagramm
	eTR_9	
	eTR_9	
	eTR_{19}	
	eTR_{15}	
	eTR_2	

Durch die aufgeführten Transformationsregeln entsteht das konzeptionelle Transformationsmodell in den Abbildungen 56 und 57. Werden diese mit dem Quellmodell verglichen, ist festzustellen: Die Ablauflogik ist beibehalten und es kann vom gleichen Geschäftsprozess gesprochen werden.

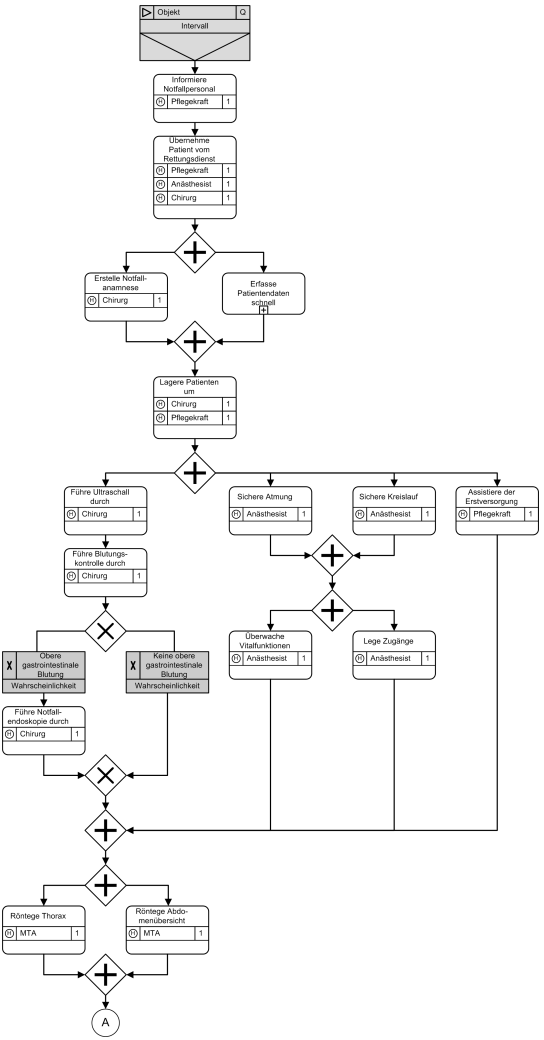


Abbildung 56: Konzeptionelles Transformationsmodell –Chirurgischer Notfall – Teil 1

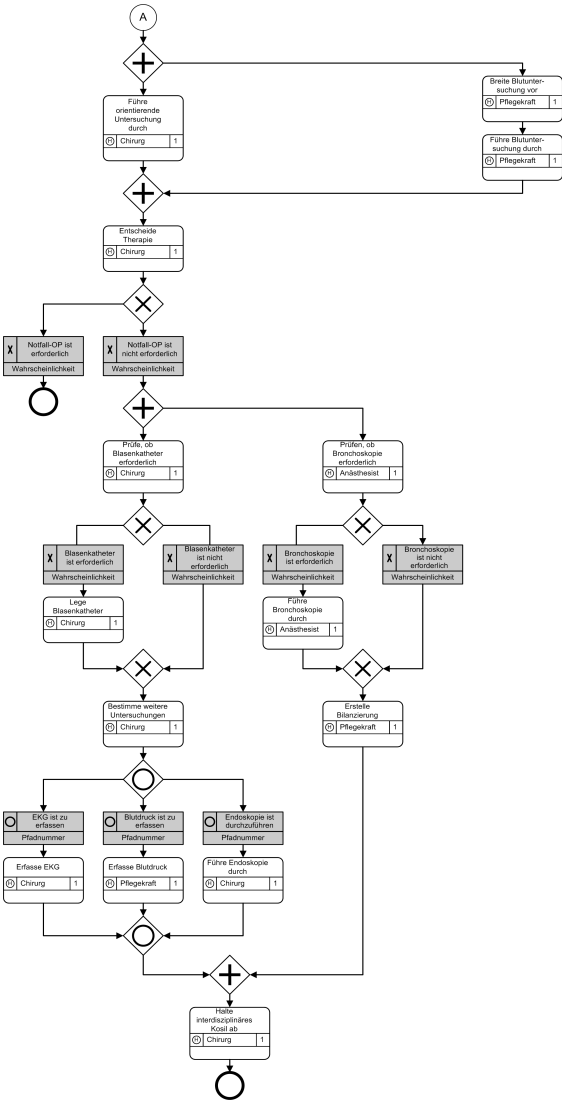


Abbildung 57: Konzeptionelles Transformationsmodell –Chirurgischer Notfall – Teil 2

Voruntersuchung

Der zweite für die Normalisierung vorgesehene Geschäftsprozess entstammt ebenfalls aus einer Geschäftsprozessanalyse und wurde im Rahmen der Diplomarbeit von Jähnke (2009) modelliert. Der Prozess beschreibt eine Voruntersuchung einer Augenambulanz in einem Krankenhaus. Im Rahmen der Diplomarbeit von Nitsch (2010) wurden die modellierten Modelle aufgegriffen und für ein Simulationsmodell verwendet. Das Geschäftsprozessmodell der Voruntersuchung ist in den Abbildungen 58, 59 und 60 dargestellt.

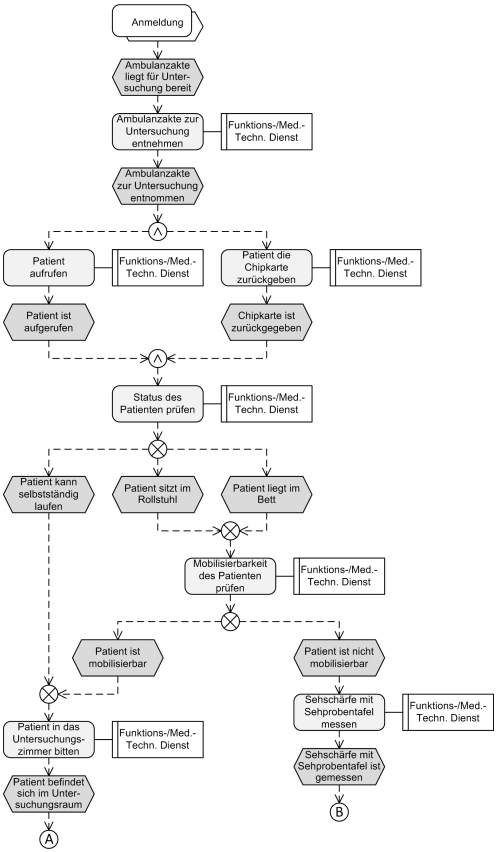


Abbildung 58: eEPK – Voruntersuchung – Teil 1

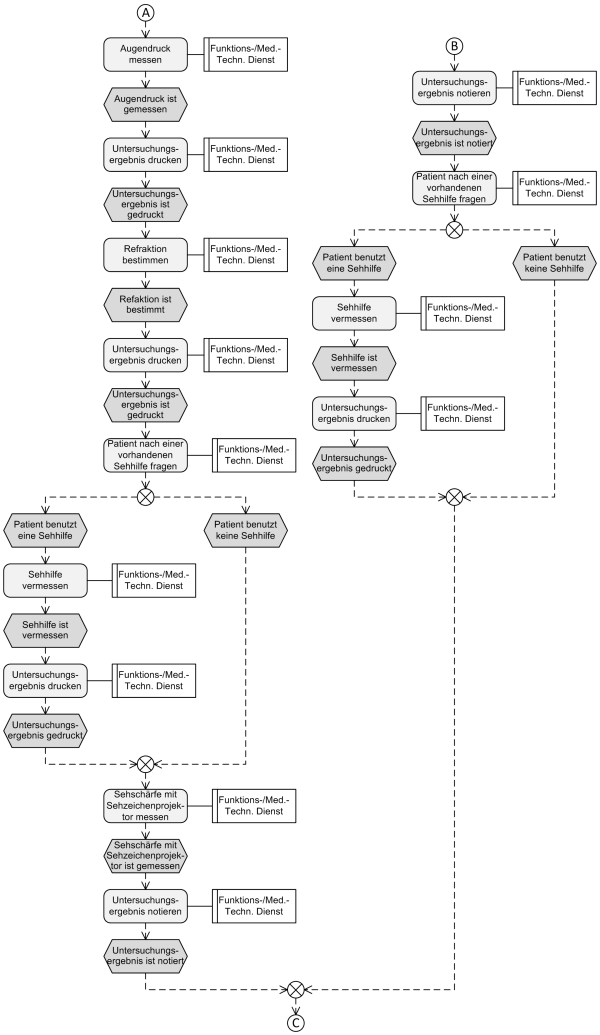


Abbildung 59: eEPK – Voruntersuchung – Teil 2

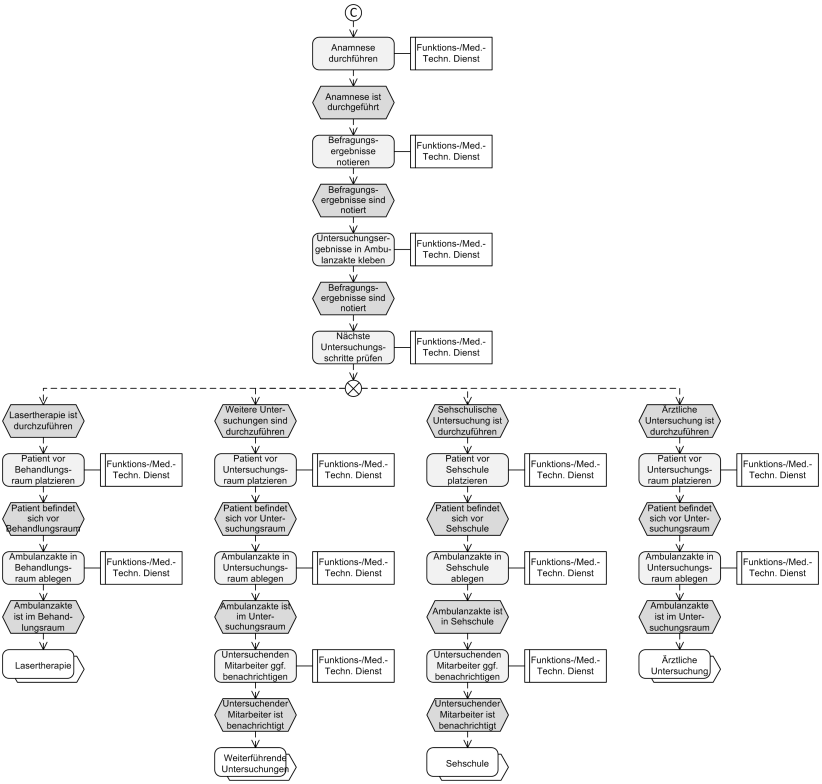


Abbildung 60: eEPK – Voruntersuchung – Teil 3

Anwendung der Vorbereitungsregeln

Die Vorbereitungsregeln entfernen auch bei diesem Geschäftsprozessmodell die Ereignisse, die nicht für die weitere Transformation benötigt werden. Aufgrund des stark linearen Ablaufs wird lediglich die Vorbereitungsregel ePR_1 angewendet. Gleichnamige Ereignisse sind in der Tabelle 41 nicht mehrfach aufgeführt.

Tabelle 41: Anwendung der eEPK Vorbereitungsregeln auf die Voruntersuchung

Ereignis	Vorbereitungsregel	Aktion
Ambulanzakte ist zur Untersuchung entnommen	ePR_1	Ereignis entfernt
Patient ist aufgerufen	ePR_1	Ereignis entfernt
Chipkarte ist zurückgegeben	ePR_1	Ereignis entfernt
Patient befindet sich im Untersuchungsraum	ePR_1	Ereignis entfernt
Sehschärfe mit Sehprobentafel ist gemessen	ePR_1	Ereignis entfernt
Augendruck ist gemessen	ePR_1	Ereignis entfernt
Untersuchungsergebnis ist gedruckt	ePR_1	Ereignis entfernt
Refraktion ist bestimmt	ePR_1	Ereignis entfernt
Sehhilfe ist vermessen	ePR_1	Ereignis entfernt
Sehschärfe mit Sehzeichenprojektor ist gemessen	ePR_1	Ereignis entfernt
Untersuchungsergebnis ist notiert	ePR_1	Ereignis entfernt
Anamnese ist durchgeführt	ePR_1	Ereignis entfernt
Befragungsergebnisse sind notiert	ePR_1	Ereignis entfernt
Patient befindet sich vor Behandlungsraum	ePR_1	Ereignis entfernt
Patient befindet sich vor Untersuchungsraum	ePR_1	Ereignis entfernt
Ambulanzakte ist im Untersuchungsraum	ePR_1	Ereignis entfernt
Patient befindet sich vor Sehschule	ePR_1	Ereignis entfernt
Ambulanzakte ist in Sehschule	ePR_1	Ereignis entfernt

Nach der mehrfachen Anwendung der Vorbereitungsregel ePR_1 entsteht das in den Abbildungen 61 und 62 dargestellte vorbereitete Geschäftsprozessmodell.

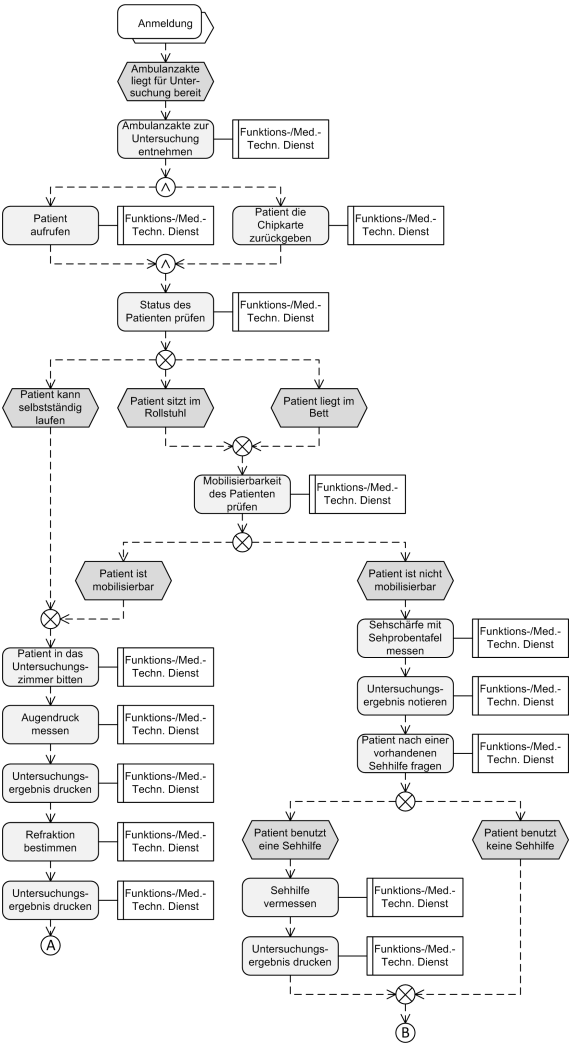


Abbildung 61: eEPK – Voruntersuchung – vorbereitet – Teil 1

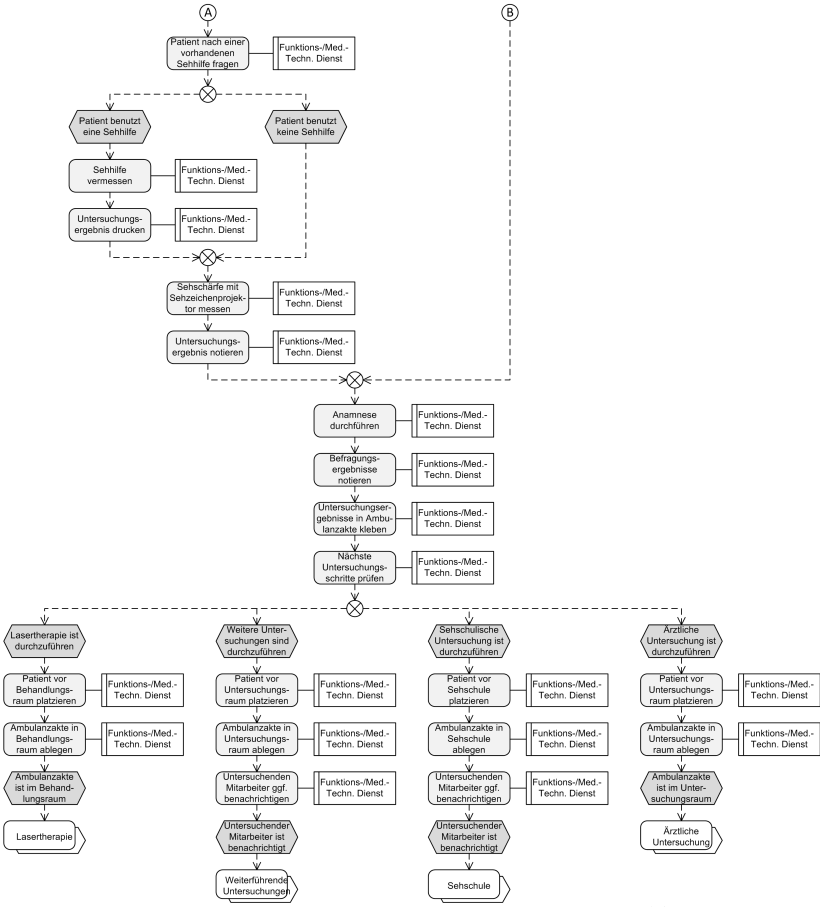
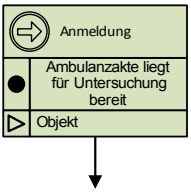
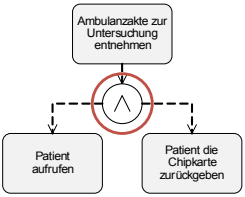
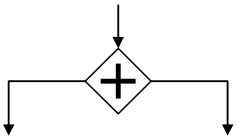
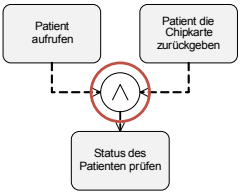
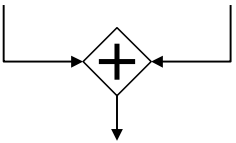
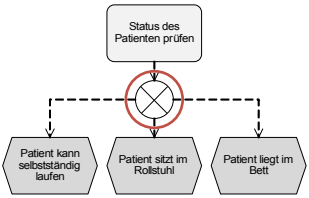
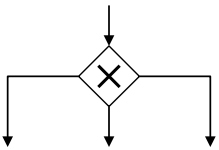
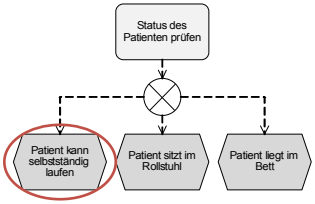
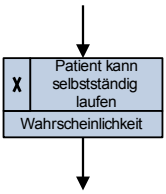


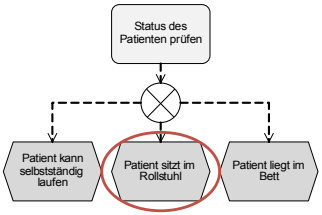
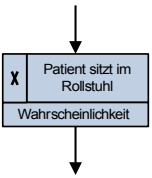
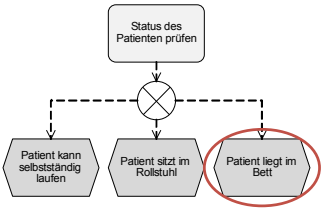
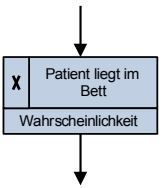
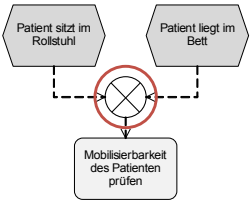
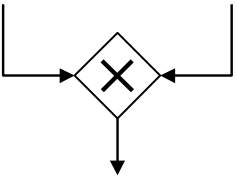
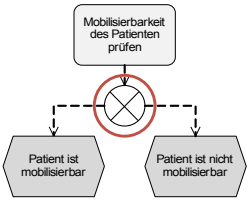
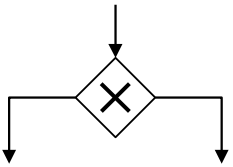
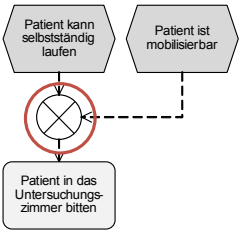
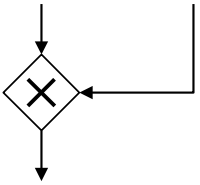
Abbildung 62: eEPK – Voruntersuchung – vorbereitet – Teil 2

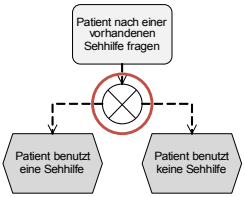
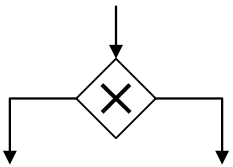
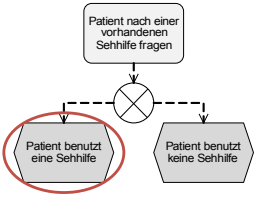
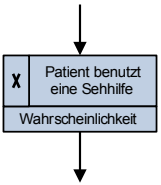
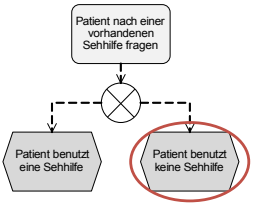
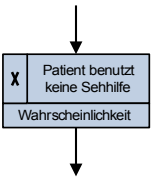
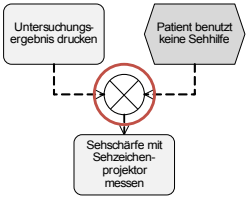
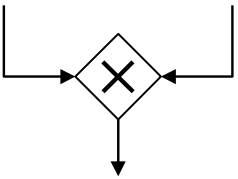
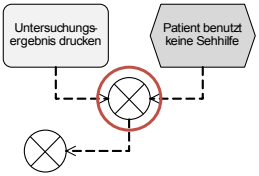
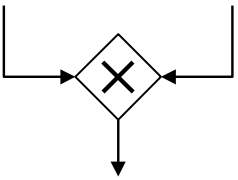
Anwendung der Transformationsregeln

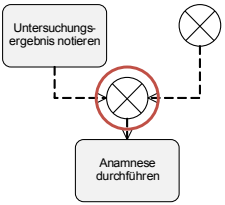
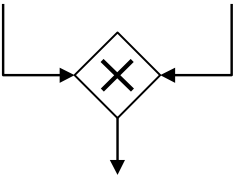
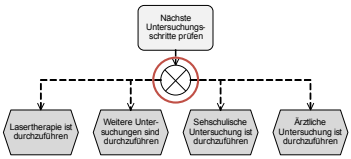
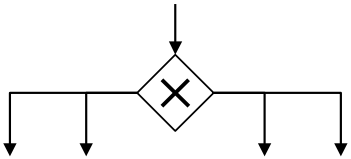
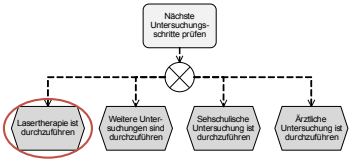
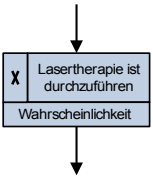
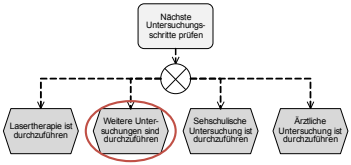
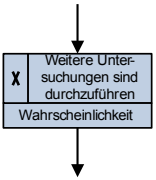
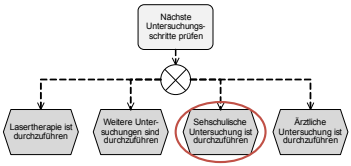
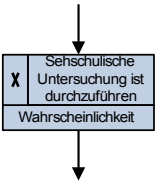
Auf die vorbereitete eEPK werden die Transformationsregeln angewendet, die in Tabelle 42 aufgelistet sind. Da keine Funktion auf einen anderen Prozess verweist, wird auf alle Funktionen die Transformationsregel eTR_{12} angewendet. Die Bezeichnung wird jeweils als Tätigkeit verwendet und als Ressource erhält jede Aktivität den „Funktions-/Med.-Techn. Dienst“.

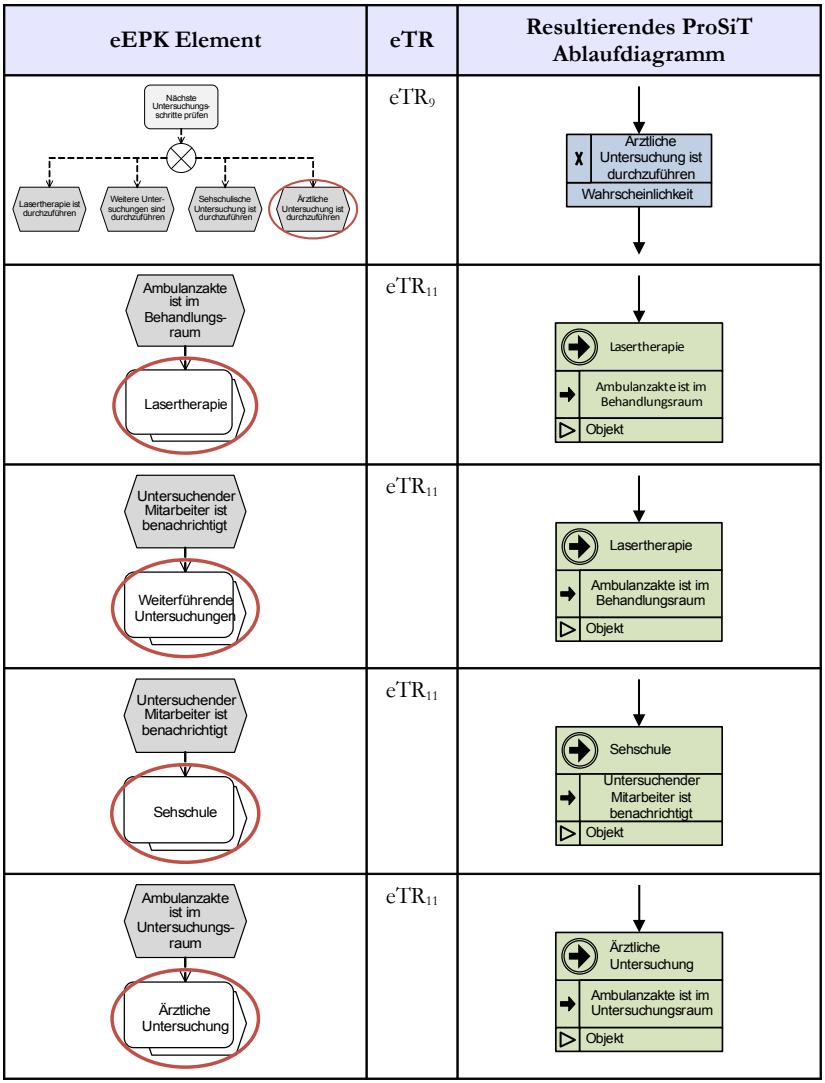
Tabelle 42: Anwendung der eEPK Transformationsregeln auf die Voruntersuchung

eEPK Element	eTR	Resultierendes ProSiT Ablaufdiagramm
	eTR_{10}	
	eTR_{14}	
	eTR_{15}	
	eTR_{18}	
	eTR_{9}	

eEPK Element	eTR	Resultierendes ProSiT Ablaufdiagramm
	eTR_9	
	eTR_9	
	eTR_{19}	
	eTR_{18}	
	eTR_{19}	

eEPK Element	eTR	Resultierendes ProSiT Ablaufdiagramm
	$e'TR_{18}$	
	$e'TR_{9}$	
	$e'TR_{9}$	
	$e'TR_{19}$	
	$e'TR_{19}$	

eEPK Element	eTR	Resultierendes ProSiT Ablaufdiagramm
	eTR_{19}	
	eTR_{18}	
	eTR_{9}	
	eTR_{9}	
	eTR_{9}	



Das resultierende konzeptuelles ProSiT Ablaufdiagramm ist in den Abbildungen 63 und 64 dargestellt.

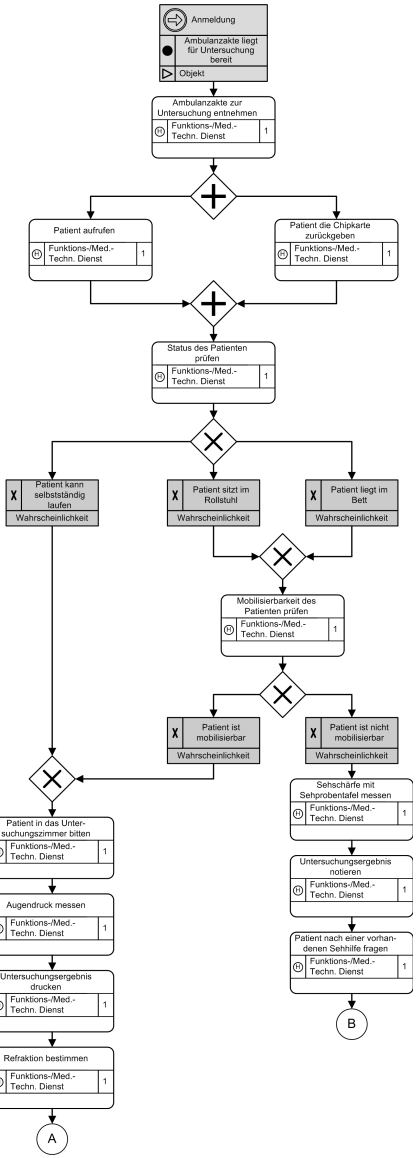


Abbildung 63: Transformationsmodell – Voruntersuchung – konzeptuell – Teil 1

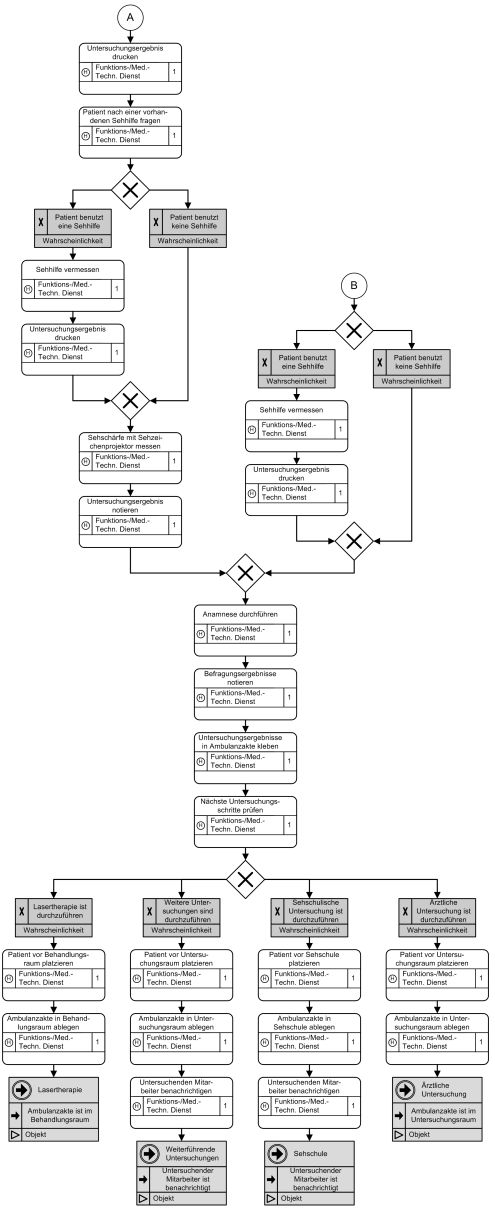


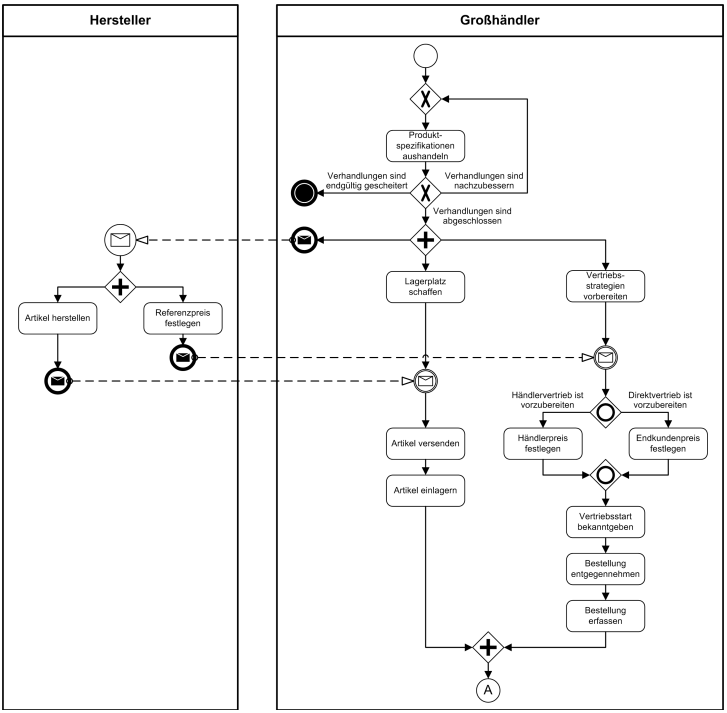
Abbildung 64: Transformationsmodell – Voruntersuchung – konzeptuell – Teil 2

4.2.2 BPMN als Quellmodell

Die Evaluation der BPMN Transformationsregeln wird zunächst mit dem Produkteinführungsprozess und anschließend mit dem Einkaufsprozess vollzogen.

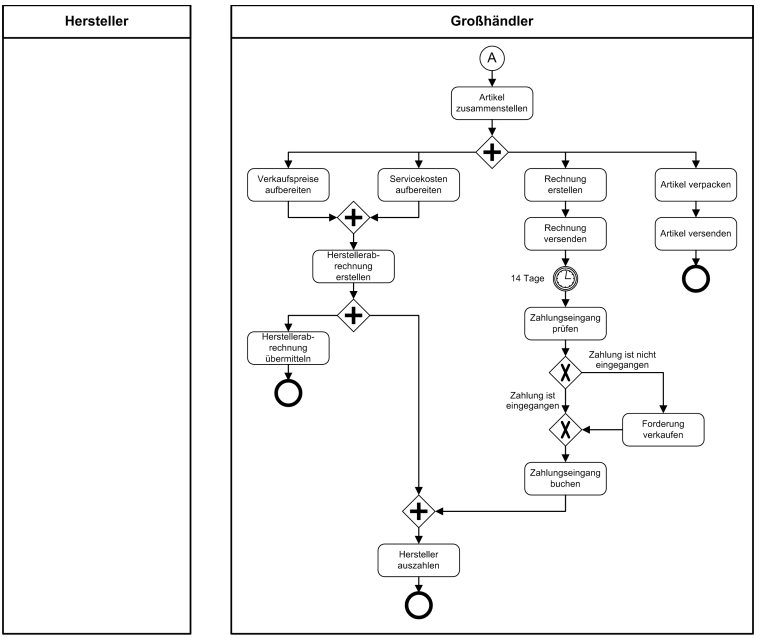
Produkteinführung

Der Prozess der Produkteinführung wurde auf der Grundlage des Artikels von Overhage et al. (2011) modelliert, um einen Vergleich der Transformationsregeln durchführen zu können. Das Geschäftsprozessmodell ist in den Abbildungen 65 und 66 dargestellt.



Nach Overhage et al. (2011, S. 749, 750)

Abbildung 65: BPMN – Produkteinführung – Teil 1



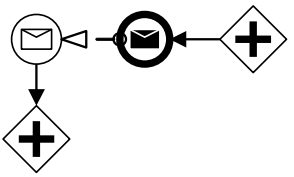
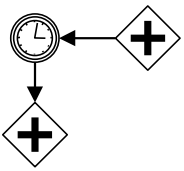
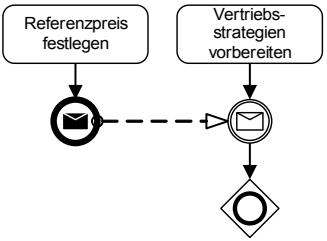
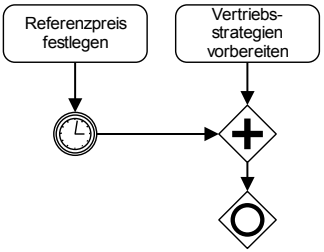
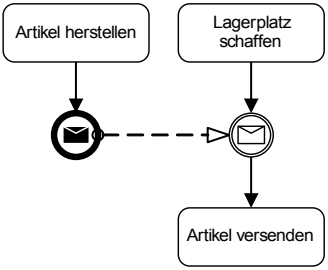
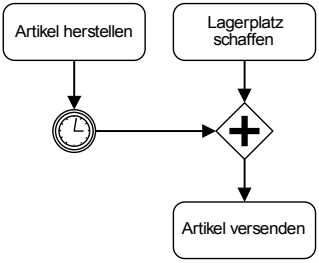
Nach Overhage et al. (2011, S. 749, 750)

Abbildung 66: BPMN – Produkteinführung – Teil 2

Anwendung der Vorbereitungsregeln

Da das BPMN Modell gemäß der BPMN Spezifikationen modelliert wurde, müssen nur drei Vorbereitungsregeln angewendet werden. Diese betreffen die Nachrichtenflüsse in Abbildung 65. In Tabelle 43 sind die drei angewendeten Regeln dargestellt. Zum Einsatz kommt jeweils die Vorbereitungsregel bPR_{2_1}; einmal mit der Ergänzungsregel bPR_{N_1} und zwei Mal mit bPR_{N_2}. Durch die Regeln wird nur der erste Teil des BPMN Modells verändert, der in Abbildung 67 dargestellt ist. Der zweite Teil entspricht dem Ausgangsmodell in Abbildung 66.

Tabelle 43: Anwendung der BPMN Vorbereitungsregeln auf die Produkteinführung

BPMN Element	bPR	Resultierendes BPMN Element
	bTR_{2_1} bPR_{N_1}	
	bTR_{2_2} bPR_{N_1}	
	bTR_{2_2} bPR_{N_1}	

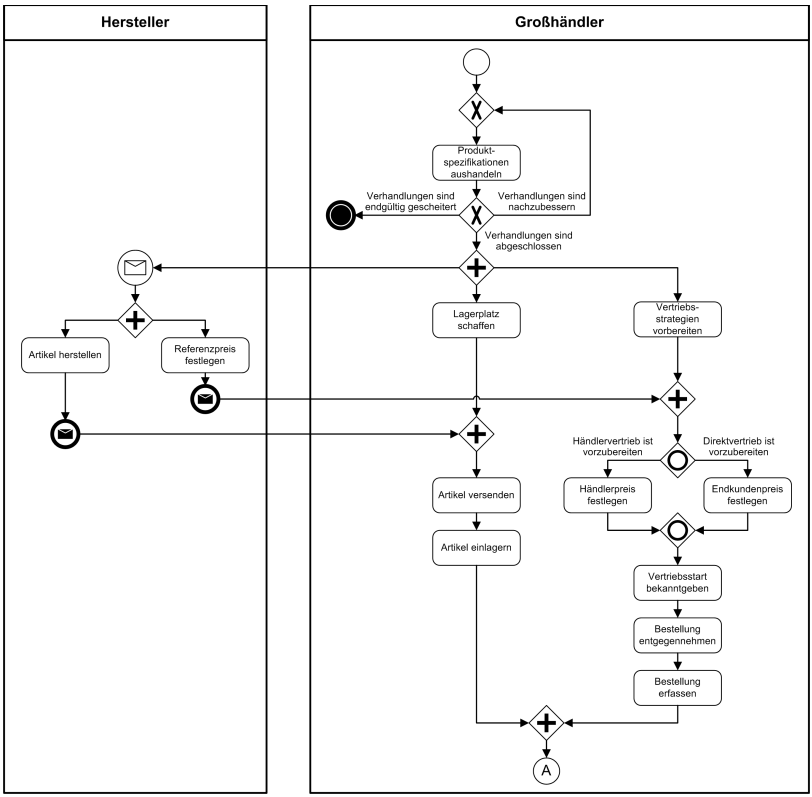


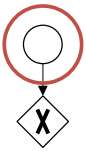
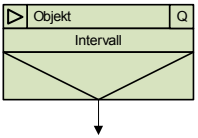
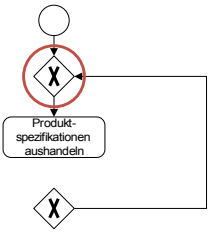
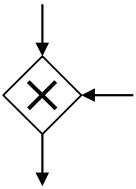
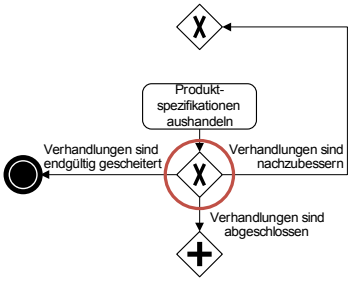
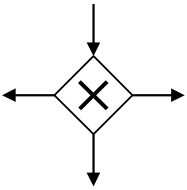
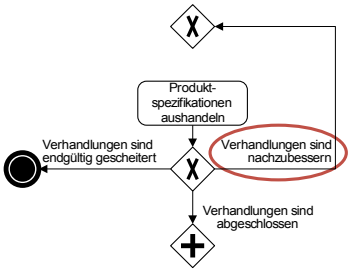
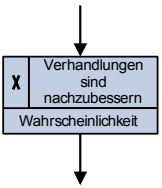
Abbildung 67: BPMN – Produkteinführung – vorbereitet – Teil 1

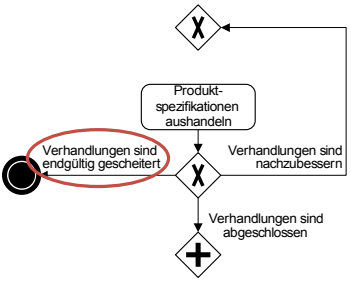
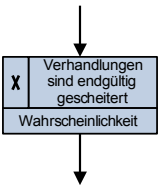
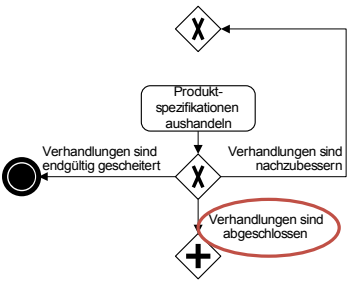
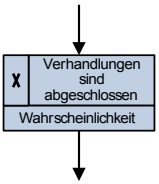
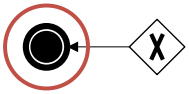
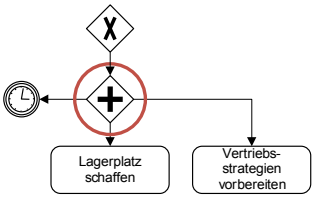
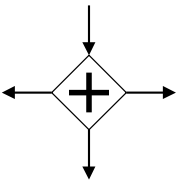
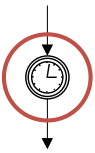
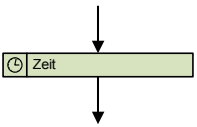
Anwendung der Transformationsregeln

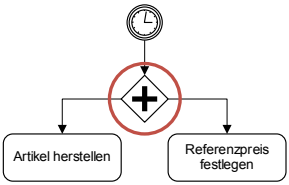
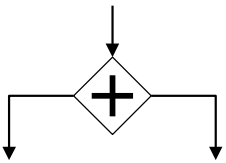
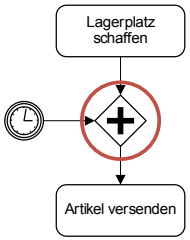
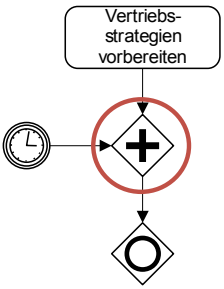
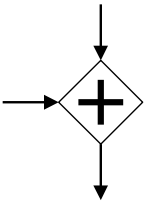
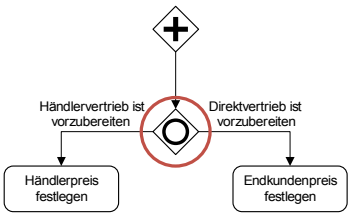
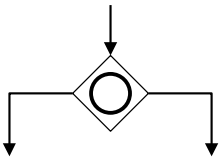
Auf das vorbereitete BPMN Modell werden anschließend die Transformationsregeln angewendet. Wie bei der Evaluation der eEPK Transformationsregeln werden die Aufgaben nicht einzeln aufgeführt. Auf alle Aufgaben wird die Transformationsregel bTR_{13} angewendet. Da sich die Aufgaben nur in einem Pool und nicht in einer Lane befinden, werden die resultierenden Aktivitäten nicht mit Ressourcen versehen. Lediglich die Bezeichnung der Aufgaben für die Tätigkeit der Aktivität wird verwendet. Die Transformationsregeln, die aus die

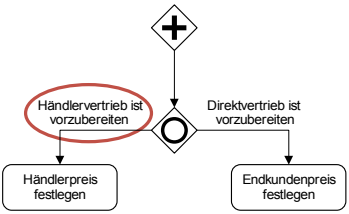
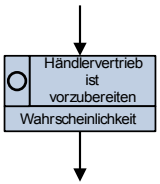
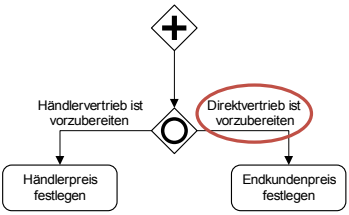
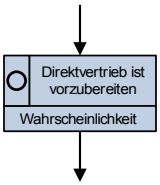
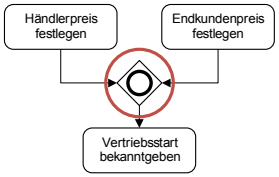
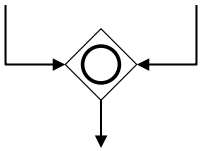
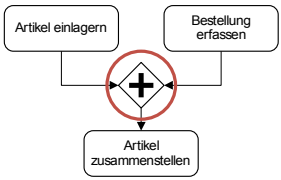
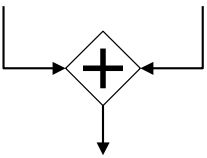
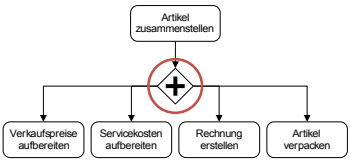
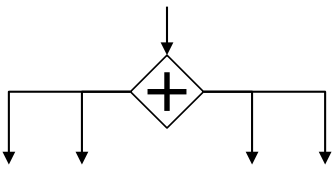
restlichen Elemente angewendet werden, sind in der nachfolgenden Tabelle 41 aufgeführt.

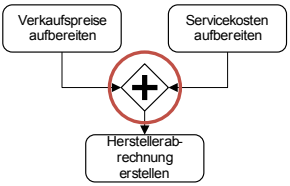
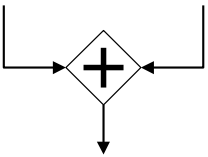
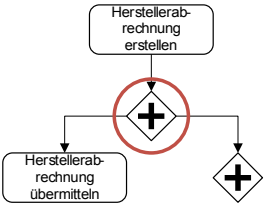
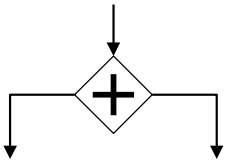
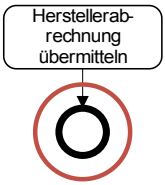
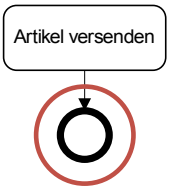
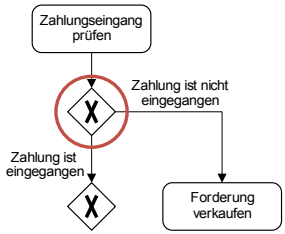
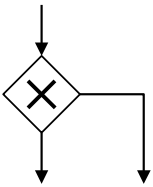
Tabelle 44: Anwendung der BPMN Transformationsregeln auf die Produkteinführung

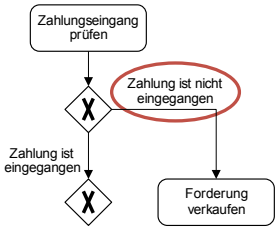
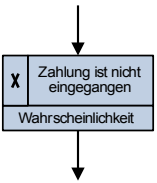
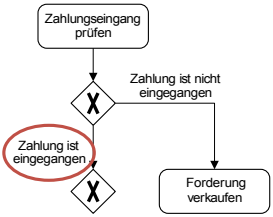
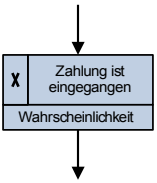
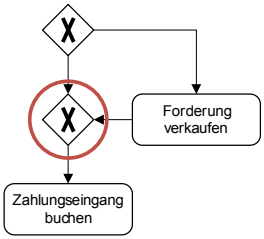
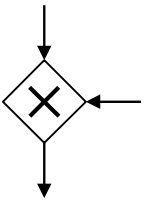
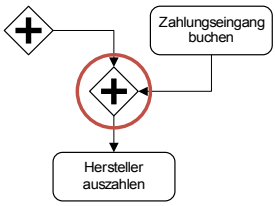
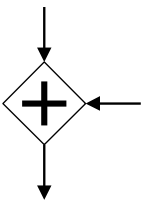
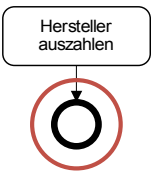
BPMN Element	bTR	Resultierendes ProSiT Ablaufdiagramm
	bTR_1	
	bTR_{10}	
	bTR_9	
	bTR_9	

BPMN Element	bTR	Resultierendes ProSiT Ablaufdiagramm
	bTR$_9$	
	bTR$_9$	
	bTR$_3$	
	bTR$_5$	
	bTR$_4$	

BPMN Element	bTR	Resultierendes ProSiT Ablaufdiagramm
	bTR_6	
	bTR_5	
	bTR_6	
	bTR_7	

BPMN Element	bTR	Resultierendes ProSiT Ablaufdiagramm
	bTR_7	
	bTR_7	
	bTR_8	
	bTR_6	
	bTR_5	

BPMN Element	bTR	Resultierendes ProSiT Ablaufdiagramm
	bTR_6	
	bTR_5	
	bTR_2	
	bTR_2	
	bTR_9	

BPMN Element	bTR	Resultierendes ProSiT Ablaufdiagramm
	bTR_9	
	bTR_9	
	bTR_{10}	
	bTR_6	
	bTR_2	

Durch Transformation entsteht das in den Abbildungen 68 und 69 dargestellte konzeptuelle ProSiT Ablaufdiagramm. Bei einem ersten Vergleich mit dem aus dem eEPK Modell erzeugten Ablaufdiagramm kann der gleiche Ablauf festgestellt werden. Wird der Modellbegriff herangezogen, dann kann geschlussfolgert werden, dass das gleiche Original beschrieben wird.

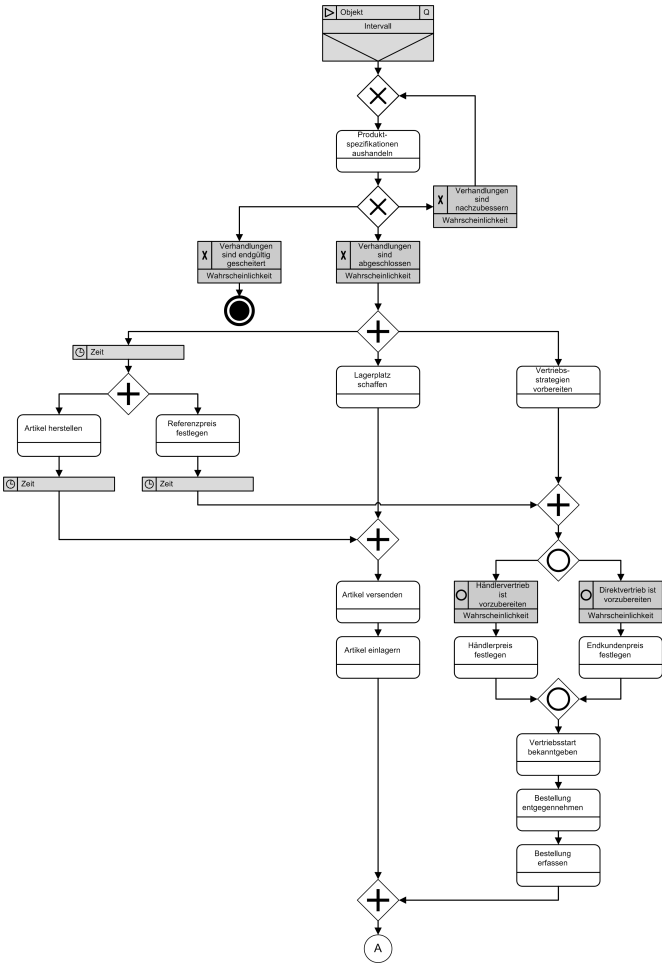


Abbildung 68: Transformationsmodell –Produkteinführung (BPMN) – konzeptuell – Teil 1

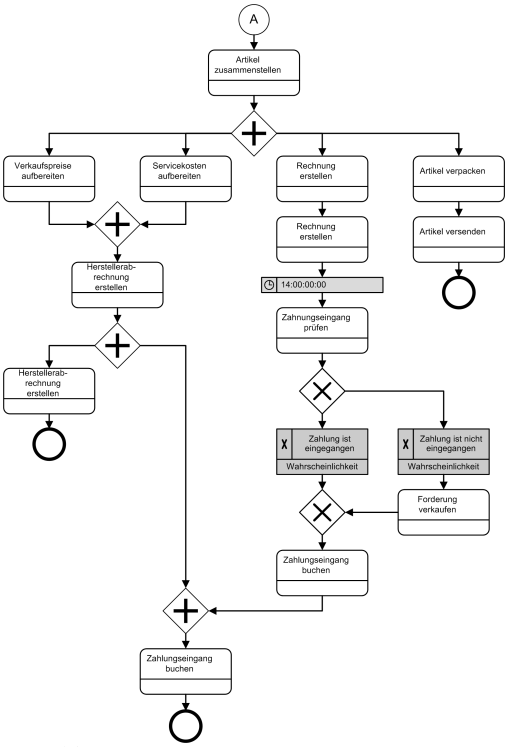


Abbildung 69: Transformationsmodell –Produkteinführung (BPMN) – konzeptuell – Teil 2

Einkaufsprozess

Der Einkaufsprozess betrachtet ausschließlich den Ablauf des Einkaufs. Daher wird nach der Disposition, wenn Planaufträge erzeugt werden, der Prozessablauf nicht weiter betrachtet, sondern endet mit einem Endereignis. Der Einkaufsprozess ist in den Abbildungen 70 und 71 dargestellt.

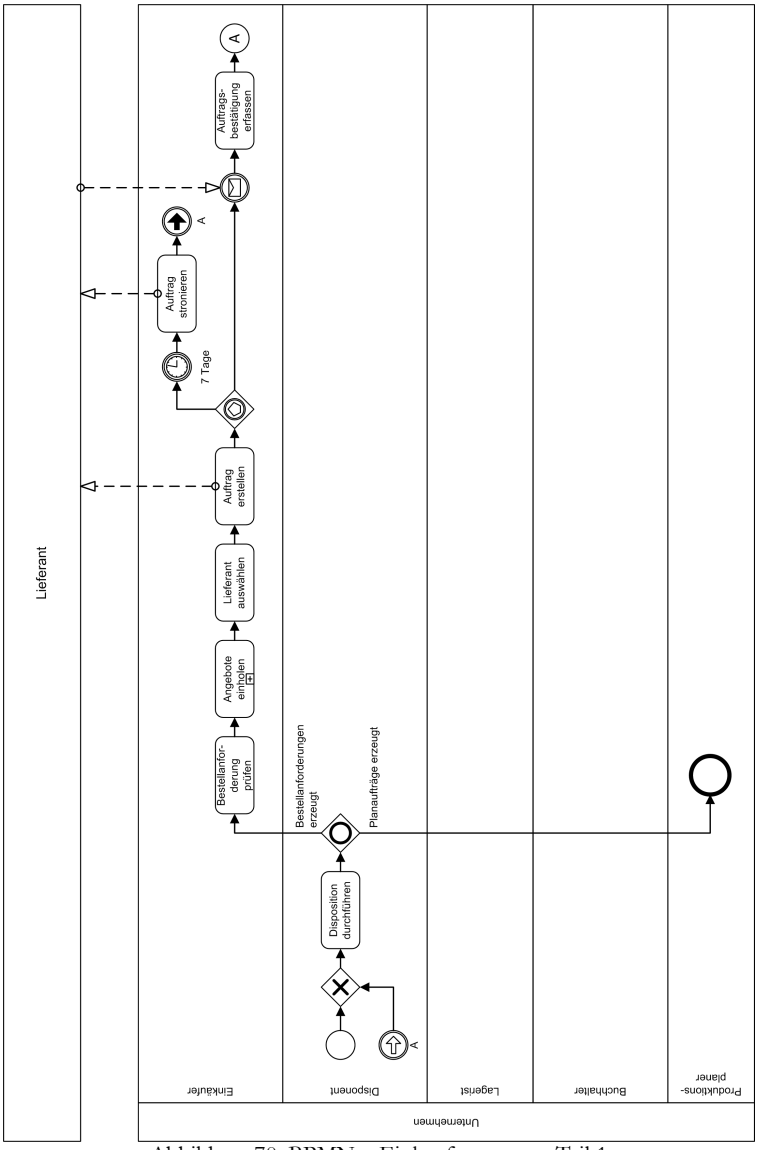


Abbildung 70: BPMN – Einkaufsprozess – Teil 1

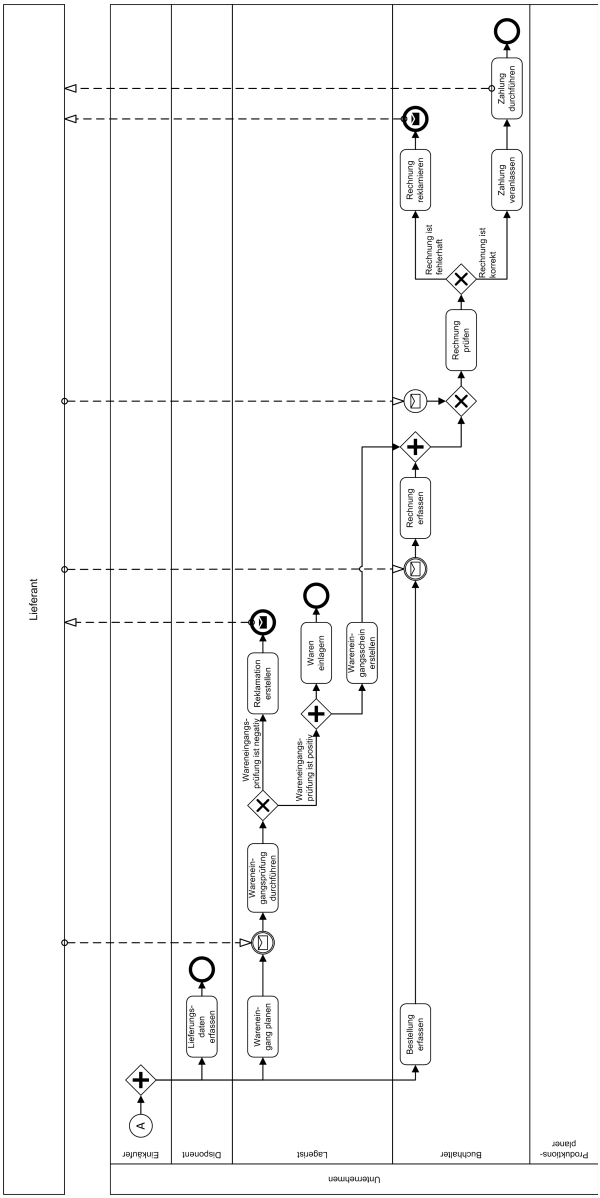
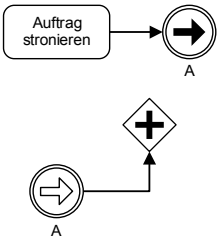
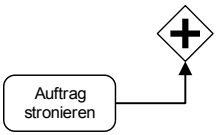
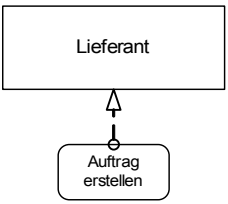
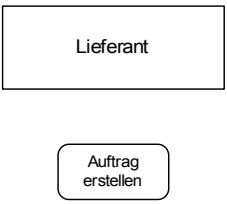
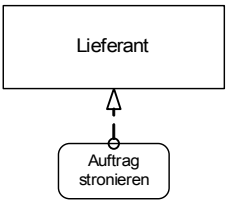
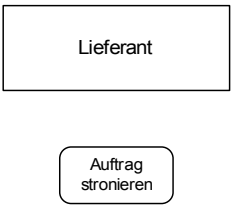


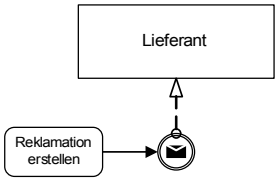
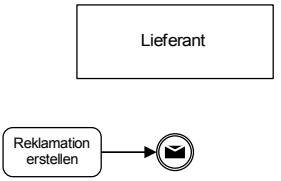
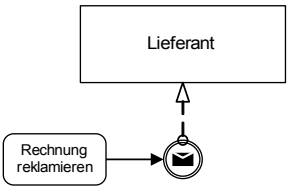
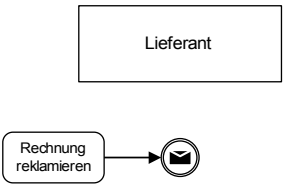
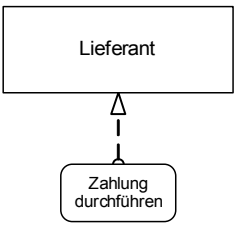
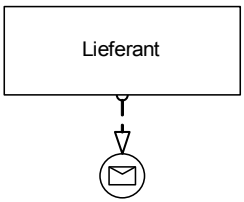
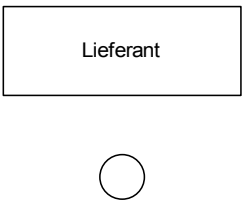
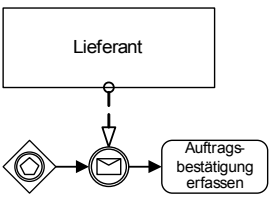
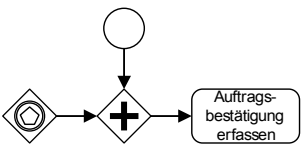
Abbildung 71: BPMN – Einkaufsprozess – Teil 2

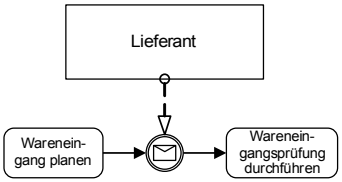
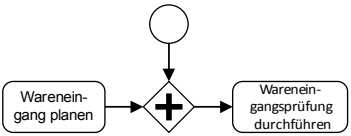
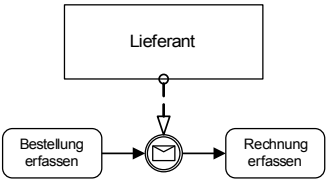
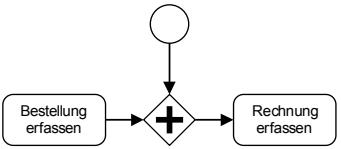
Anwendung der Vorbereitungsregeln

Auf den BPMN Prozess werden vorwiegend die Vorbereitungsregeln bPR_{1_7} sowie bPR_{2_4} in Kombination mit der Vorbereitungsregelserie bPR_N angewendet. Eine Auflistung der angewendeten Vorbereitungsregeln ist in Tabelle 45 dargestellt. Das resultierende vorbereitete BPMN Modell ist in den darauffolgenden Abbildungen 72 und 73 dargestellt.

Tabelle 45: Anwendung der BPMN Vorbereitungsregeln auf den Einkaufsprozess

BPMN Element	bPR	Resultierendes BPMN Element
	bTR_{1_6}	
	bTR_{1_7}	
	bTR_{1_7}	

BPMN Element	bPR	Resultierendes BPMN Element
	bTR_{1_7}	
	bTR_{1_7}	
	bTR_{1_7}	
	bTR_{2_1} bPR_{N_1}	
	bTR_{2_1} bPR_{N_2}	

BPMN Element	bPR	Resultierendes BPMN Element
	bTR_{2_1} bPR_{N_2}	
	bTR_{2_1} bPR_{N_2}	
	bTR_{2_6}	

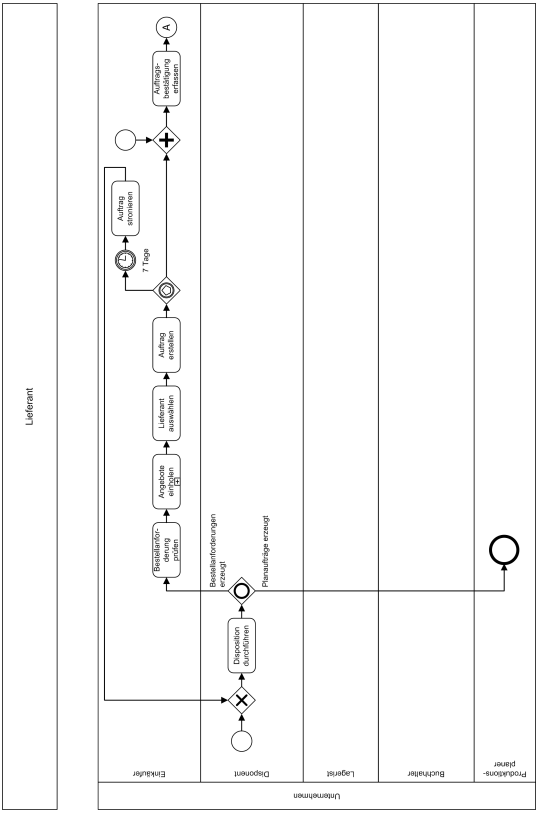


Abbildung 72: BPMN – Einkaufsprozess – vorbereitet – Teil 1

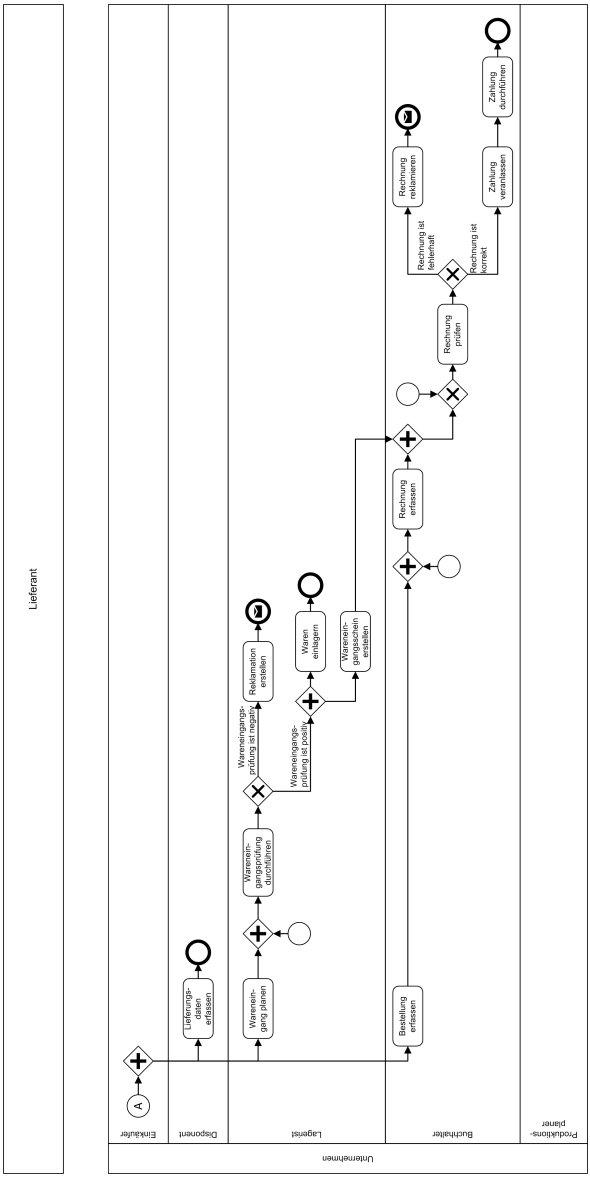
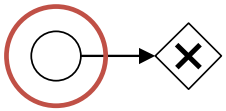
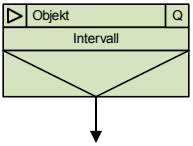
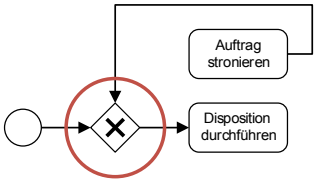
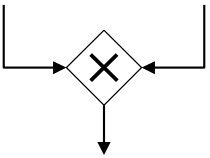
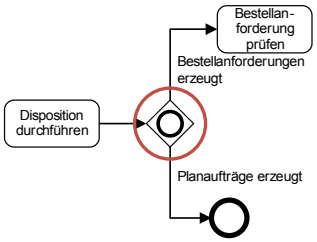
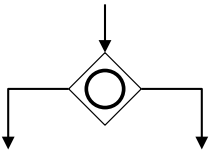


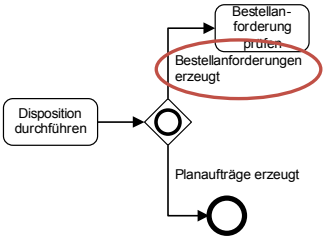
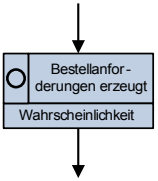
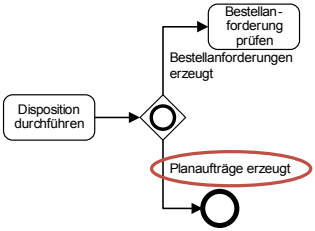
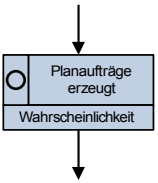
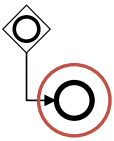
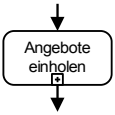
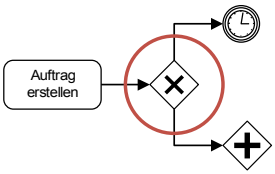
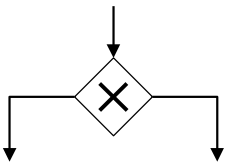
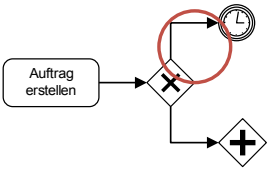
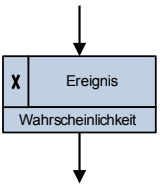
Abbildung 73: BPMN – Einkaufsprozess – vorbereitet – Teil 2

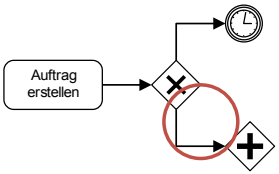
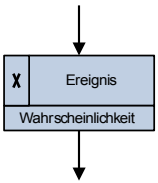
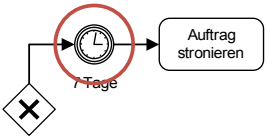
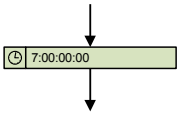
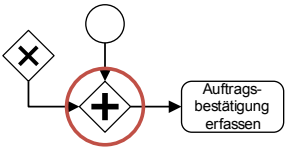
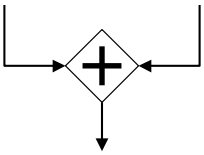
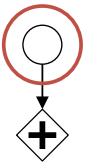
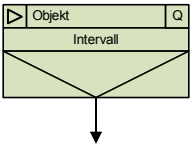
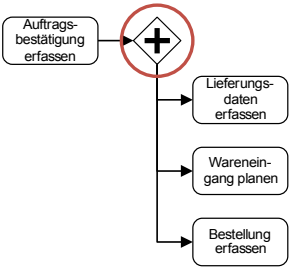
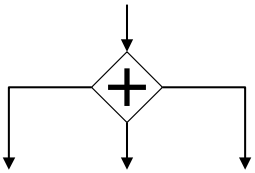
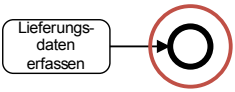
Anwendung der Transformationsregeln

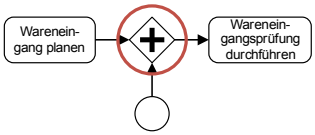
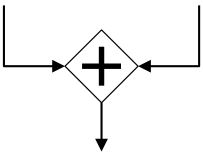
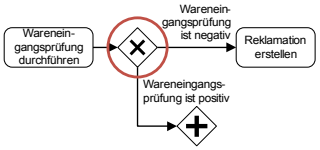
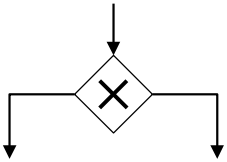
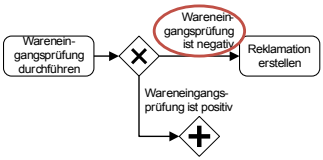
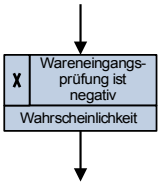
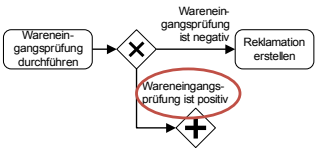
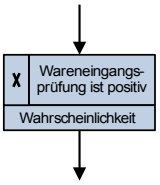
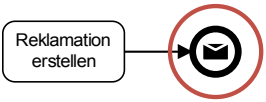
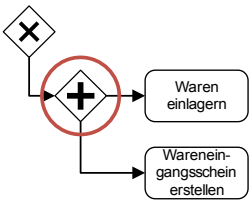
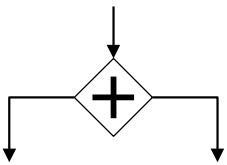
Aufbauend auf das vorbereitete Modell werden anschließend die Transformationsregeln angewendet. Aufgaben werden nachfolgenden nicht bei den Transformationsregeln aufgeführt, da auf alle die Transformationsregel bTR_{13} angewendet wird. Da sich im Gegensatz zum BPMN Prozess der Produkteinführung die Aufgaben in Lanes platziert sind, schaltet die Zusatzregel, womit die Bezeichnung der Lane als Ressource der Aktivität hinzugefügt wird. Die restlichen Transformationsregeln sind in Tabelle 46 aufgeführt.

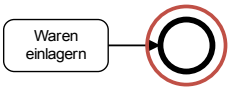
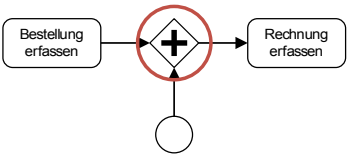
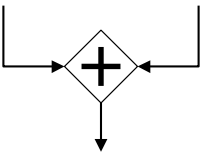
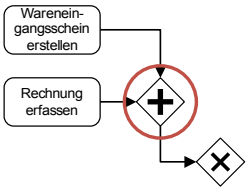
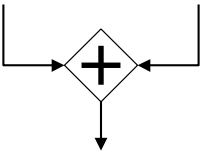
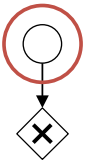
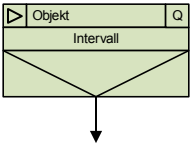
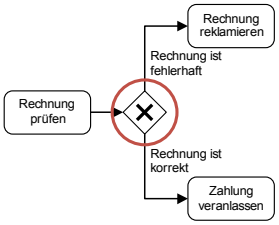
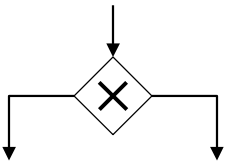
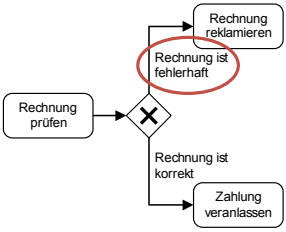
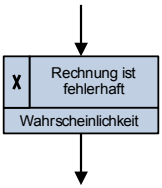
Tabelle 46: Anwendung der BPMN Transformationsregeln auf den Einkaufsprozess

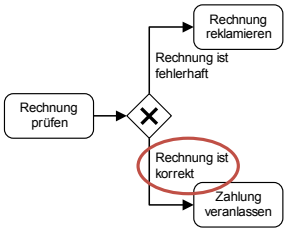
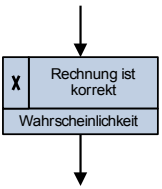
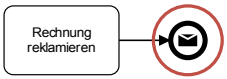
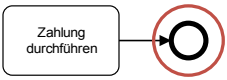
BPMN Element	bTR	Resultierendes ProSiT Ablaufdiagramm
	bTR_1	
	bTR_{10}	
	bTR_7	

BPMN Element	bTR	Resultierendes ProSiT Ablaufdiagramm
	bTR_7	
	bTR_7	
	bTR_2	
	bTR_{14}	
	bTR_9	
	bTR_9	

BPMN Element	bTR	Resultierendes ProSiT Ablaufdiagramm
	bTR_9	
	bTR_4	
	bTR_6	
	bTR_1	
	bTR_5	
	bTR_2	

BPMN Element	bTR	Resultierendes ProSiT Ablaufdiagramm
	bTR_6	
	bTR_9	
	bTR_9	
	bTR_9	
	bTR_2	
	bTR_5	

BPMN Element	bTR	Resultierendes ProSiT Ablaufdiagramm
	bTR_2	
	bTR_6	
	bTR_6	
	bTR_1	
	bTR_9	
	bTR_9	

BPMN Element	bTR	Resultierendes ProSiT Ablaufdiagramm
	bTR_9	
	bTR_2	
	bTR_2	

Die in Tabelle 46 aufgeführten sowie die Transformationsregel bTR_{13} für die Aufgaben erzeugen das in den Abbildungen 74 und 75 aufgezeigt konzeptuelle ProSiT Ablaufdiagramm. Da der Ablauf und die Logik beibehalten wurden, kann aus semantischer Sicht vom gleichen Modell gesprochen werden.

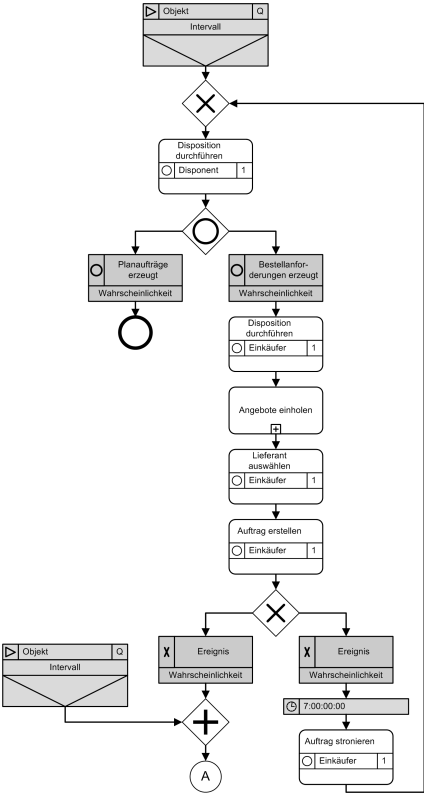


Abbildung 74: Transformationsmodell –Einkaufsprozess – konzeptuell – Teil 1

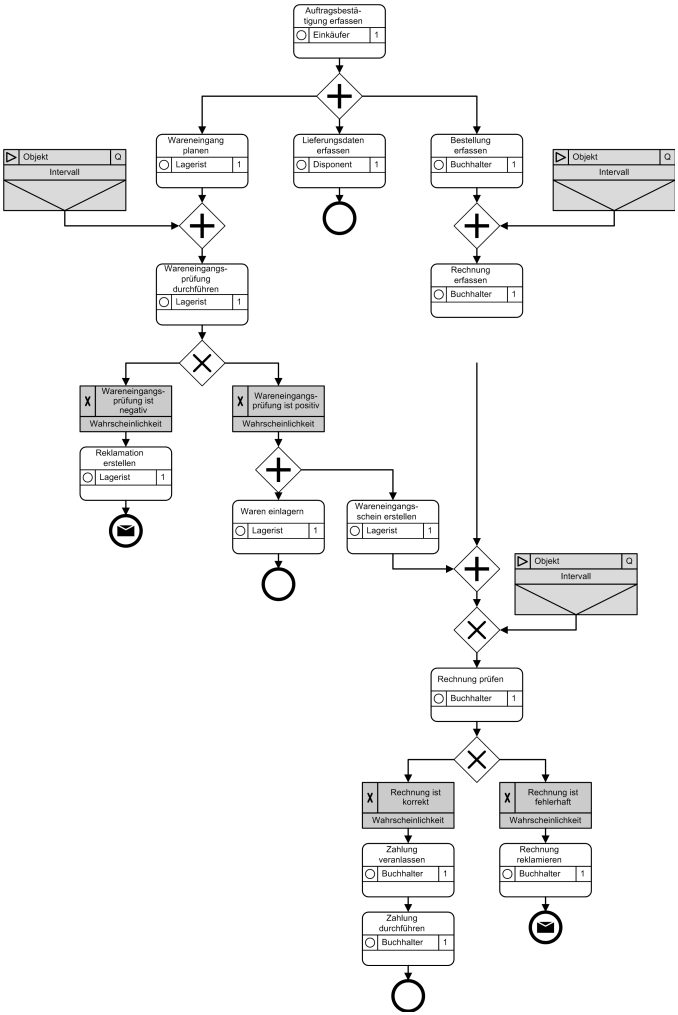


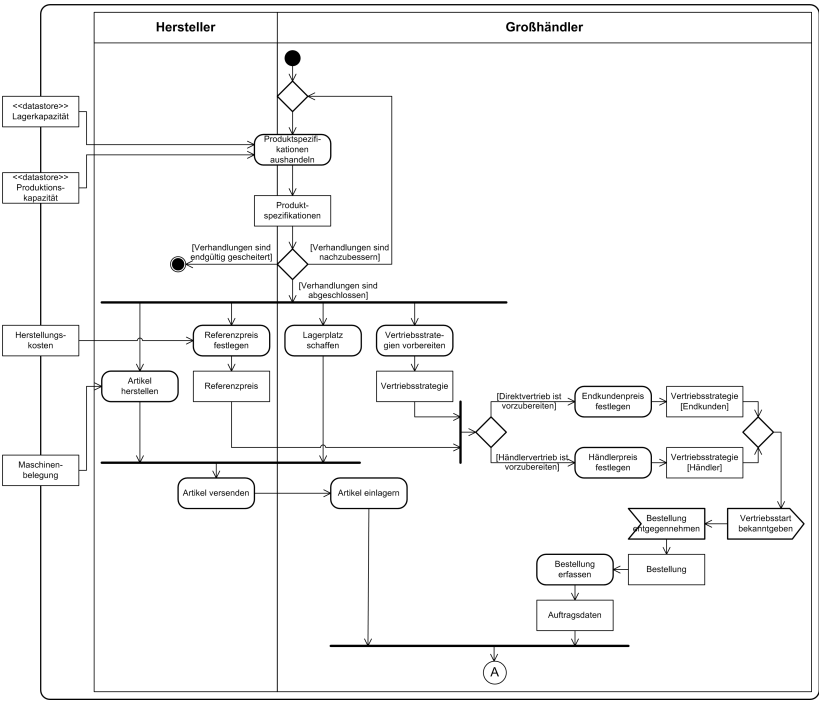
Abbildung 75: Transformationsmodell –Einkaufsprozess – konzeptuell – Teil 2

4.2.3 UML Aktivitätsdiagramm als Quellmodell

Die letzte Gruppe von Quellmodellen, die in das ProSiT Ablaufdiagramm überführt werden, sind UML Aktivitätsdiagramme. Bei diesen werden zunächst die Produkteinführung und anschließend der Ablauf der Kfz-Vermietung dargestellt.

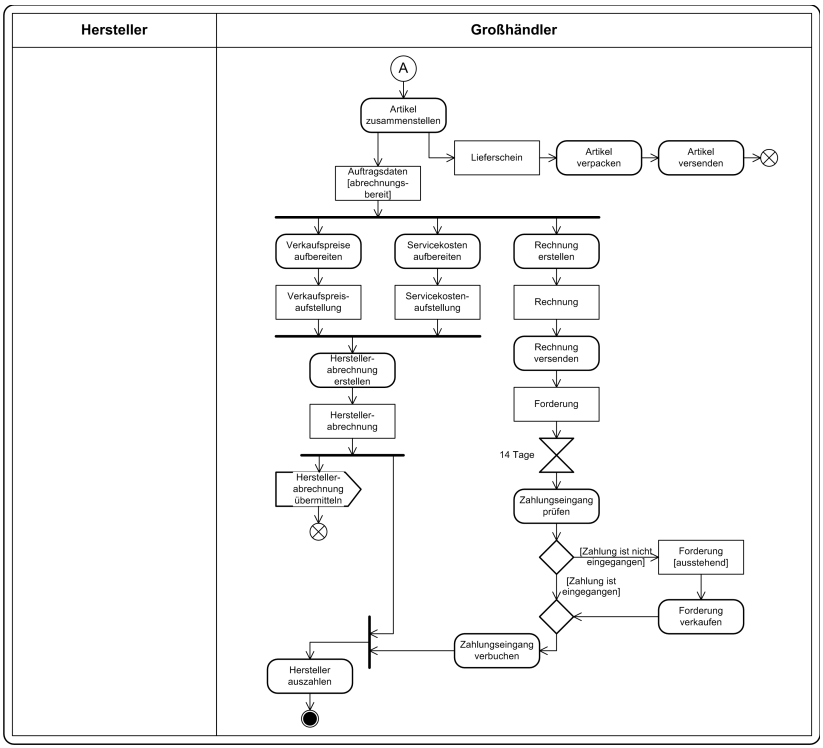
Produkteinführung

Die Produkteinführung, die mit dem UML Aktivitätsdiagramm modelliert wurde, ist in den Abbildungen 76 und 77 dargestellt.



Nach Overhage (2011, S. 749)

Abbildung 76: UML – Produkteinführung – Teil 1



Nach Overhage (2011, S. 749)

Abbildung 77: UML – Produkteinführung – Teil 2

Anwendung der Vorbereitungsregeln

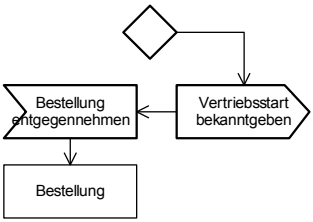
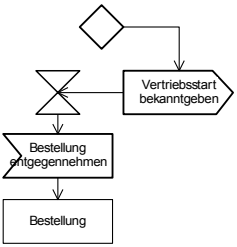
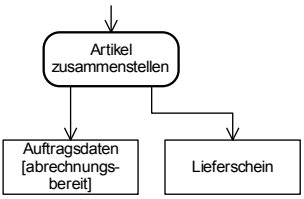
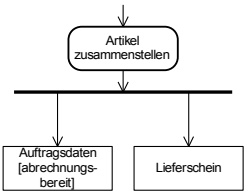
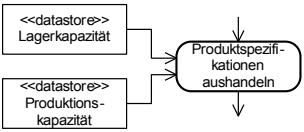
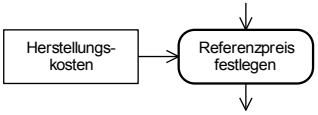
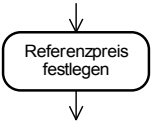
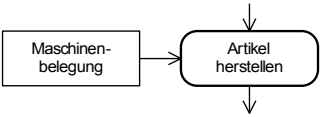
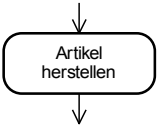
Auf das dargestellte UML Aktivitätsdiagramm werden die Vorbereitungsregeln angewendet. Hierbei werden vier Vorbereitungsregeln ausgelöst. Die Vorbereitungsregel $adPR_{2\_18}$ wird als erstes in Tabelle 47 aufgelistet, auch wenn diese nach der Vorbereitungsregelserie $adPR_1$ ausgeführt wird. Diese Regel entfernt Datenobjekte, die sich innerhalb des Prozessablaufs befinden.

Tabelle 47: Anwendung der Vorbereitungsregeln $adPR_{2\_18}$ auf die UML Produkteinführung

Datenobjekt	Vorbereitungsregel	Aktion
Produktspezifikationen	$adPR_{2\_18}$	Datenobjekt entfernt
Referenzpreis	$adPR_{2\_18}$	Datenobjekt entfernt
Vertriebsstrategie	$adPR_{2\_18}$	Datenobjekt entfernt
Vertriebsstrategie [Endkunden]	$adPR_{2\_18}$	Datenobjekt entfernt
Vertriebsstrategie [Händler]	$adPR_{2\_18}$	Datenobjekt entfernt
Bestellung	$adPR_{2\_18}$	Datenobjekt entfernt
Auftragsdaten	$adPR_{2\_18}$	Datenobjekt entfernt
Auftragsdaten [abrechnungsbereit]	$adPR_{2\_18}$	Datenobjekt entfernt
Lieferschein	$adPR_{2\_18}$	Datenobjekt entfernt
Verkaufspreisaufstellung	$adPR_{2\_18}$	Datenobjekt entfernt
Servicekostenaufstellung	$adPR_{2\_18}$	Datenobjekt entfernt
Herstellerabrechnung	$adPR_{2\_18}$	Datenobjekt entfernt
Rechnung	$adPR_{2\_18}$	Datenobjekt entfernt
Forderung	$adPR_{2\_18}$	Datenobjekt entfernt
Forderung [ausstehend]	$adPR_{2\_18}$	Datenobjekt entfernt

Weitere angewendete Vorbereitungsregeln sind $adPR_{1\_6}$, welche ein Zeitereignis einfügt, $adPR_{2\_7}$, die eine Gablung hinzufügt und $adPR_{2\_16}$, welche die Datenobjekte entfernt. Eine grafische Darstellung der Anwendung dieser drei Regeln ist in Tabelle 48 aufgeführt.

Tabelle 48: Anwendung der UML AD Vorbereitungsregeln auf die Produkteinführung

UML Element	adPR	Resultierendes UML Element
	$adPR_{1\_6}$	
	$adPR_{2\_7}$	
	$adPR_{2\_16}$	
	$adPR_{2\_16}$	
	$adPR_{2\_16}$	

Das aus diesen Vorbereitungsregeln resultierende Modell ist in Abbildungen 78 dargestellt.

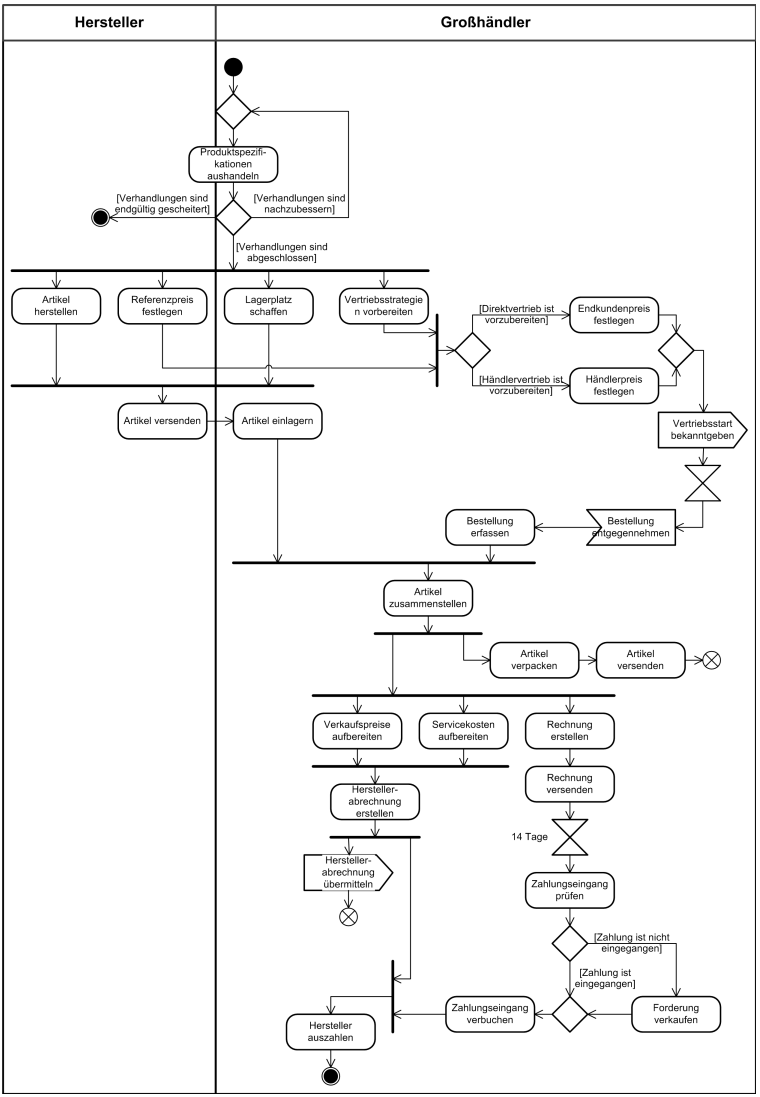
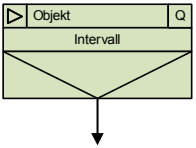
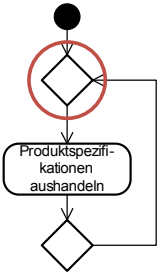
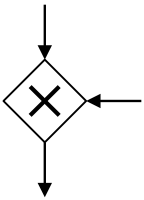


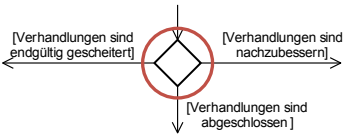
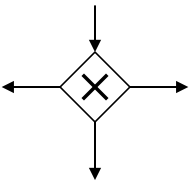
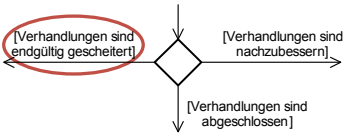
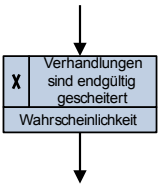
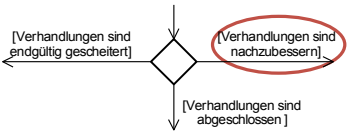
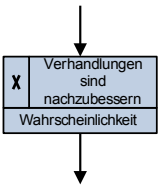
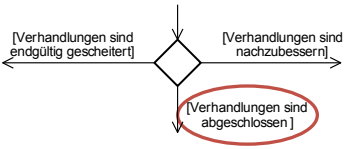
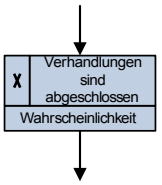
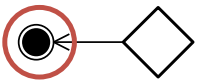
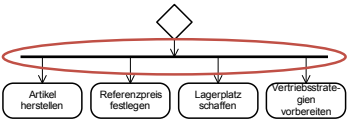
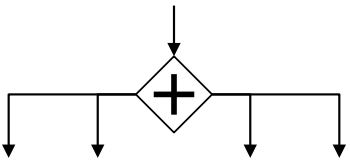
Abbildung 78: UML – Produkteinführung – vorbereitet

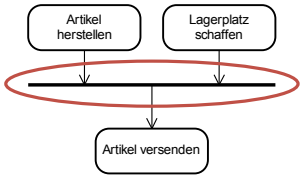
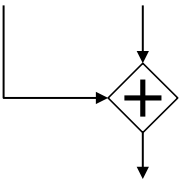
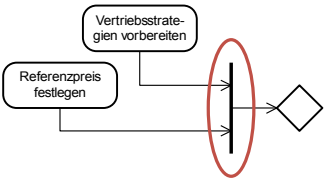
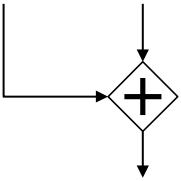
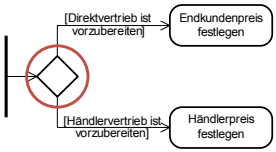
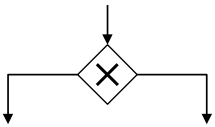
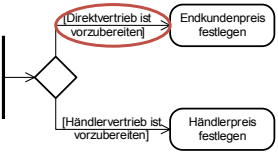
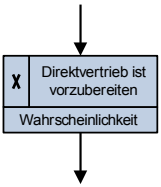
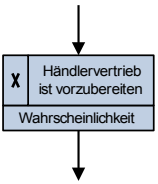
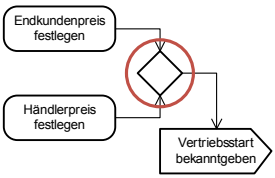
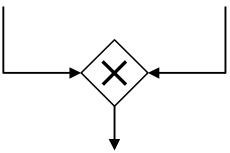
Anwendung der Transformationsregeln

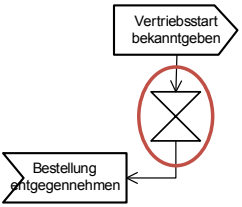
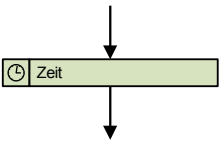
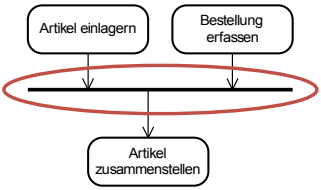
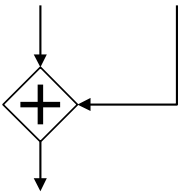
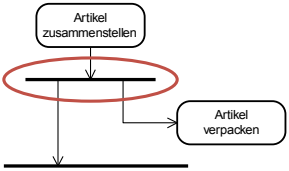
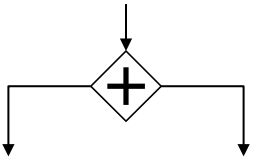
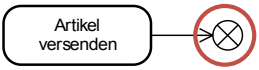
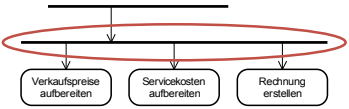
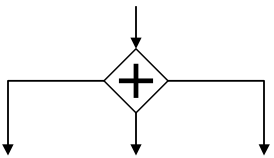
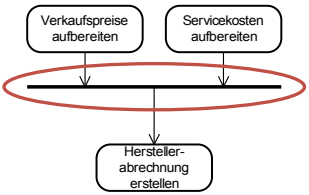
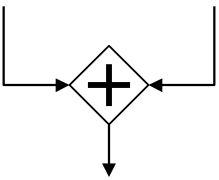
Auf das vorbereitete UML Aktivitätsdiagramm können anschließend die Transformationsregeln angewendet werden. Aktionen werden mit der Transformationsregel adTR$_7$ in Aktivitäten überführt. Die Zusatzregel der adTR$_7$ sucht die am tiefsten liegende Partition und fügt diese als Ressource der Aktivität hinzu. Beispielsweise erhält die Aktivität „Artikel herstellen“ den Hersteller und die Aktivität „Lagerplatz schaffen“ den Großhändler als Ressource. Da sich die Aktion „Produktspezifikationen aushandeln“ mehrheitlich beim Großhändler befindet und die Transformationsregel adTR$_7$ nicht vorsieht, dass eine Aktion in mehr als einer Partition eingebettet ist, erhält die resultierende Aktivität den Großhändler als Ressource. Die restlichen Elemente werden mit den Transformationsregeln überführt, die in Tabelle 49 aufgeführt sind.

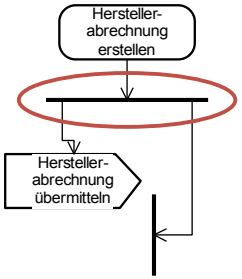
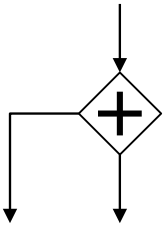
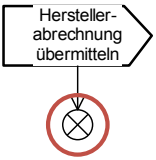
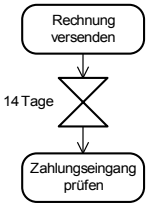
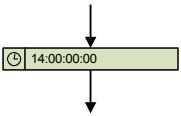
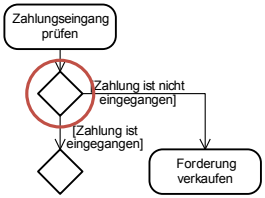
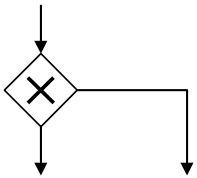
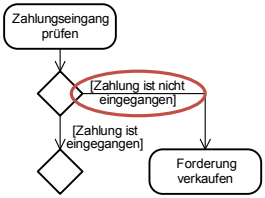
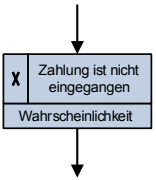
Tabelle 49: Anwendung der UML AD Transformationsregeln auf die Produkteinführung

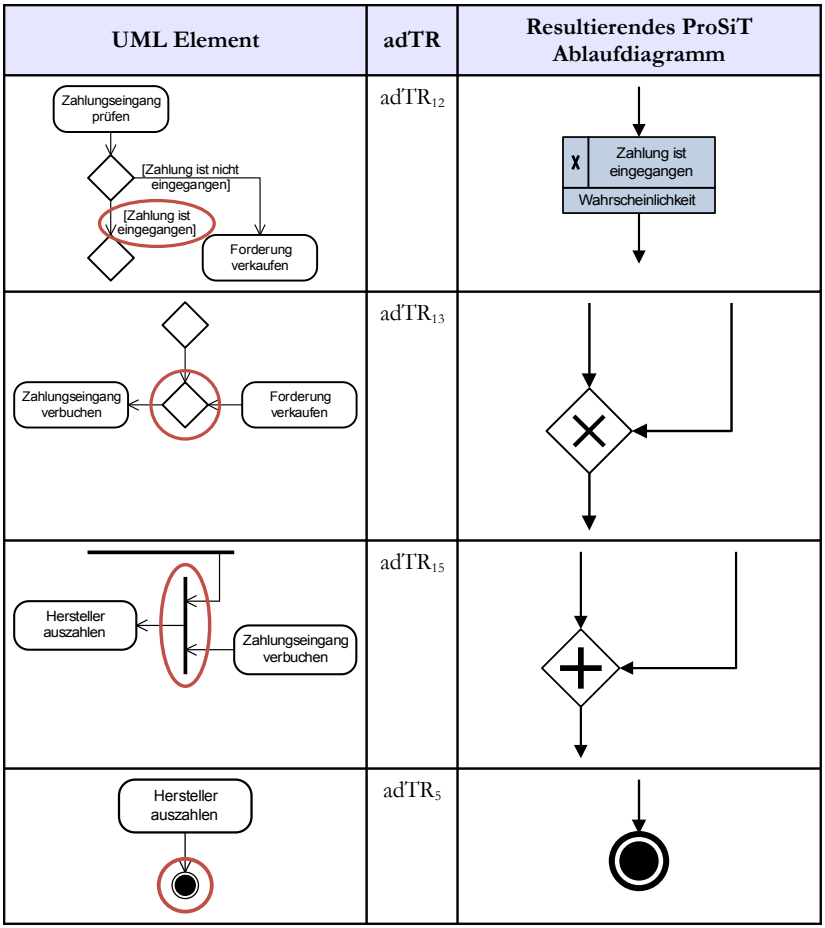
UML Element	adTR	Resultierendes ProSiT Ablaufdiagramm
	adTR$_1$	
	adTR$_{13}$	

UML Element	adTR	Resultierendes ProSiT Ablaufdiagramm
	$adTR_{12}$	
	$adTR_{12}$	
	$adTR_{12}$	
	$adTR_{12}$	
	$adTR_5$	
	$adTR_{14}$	

UML Element	adTR	Resultierendes ProSiT Ablaufdiagramm
	$adTR_{15}$	
	$adTR_{15}$	
	$adTR_{12}$	
	$adTR_{12}$	
	$adTR_{12}$	
	$adTR_{13}$	

UML Element	adTR	Resultierendes ProSiT Ablaufdiagramm
	$adTR_{10}$	
	$adTR_{15}$	
	$adTR_{14}$	
	$adTR_{3}$	
	$adTR_{14}$	
	$adTR_{15}$	

UML Element	adTR	Resultierendes ProSiT Ablaufdiagramm
	$adTR_{14}$	
	$adTR_{3}$	
	$adTR_{10}$	
	$adTR_{12}$	
	$adTR_{12}$	



Konzeptuelles Transformationsmodell

Durch die Transformationsregeln entsteht das konzeptuelles ProSiT Ablaufdiagramm in den Abbildungen 79 und 80. Aus dem Ausgangsmodell wurden lediglich Datenflüsse entfernt und die restlichen Elemente entsprechend transformiert.

Bei einem Vergleich mit den erzeugen Modellen aus der eEPK und BPMN können jedoch Unterschiede festgestellt werden. Auf diese wird im Rahmen der Diskussion der Evaluationsergebnisse eingegangen.

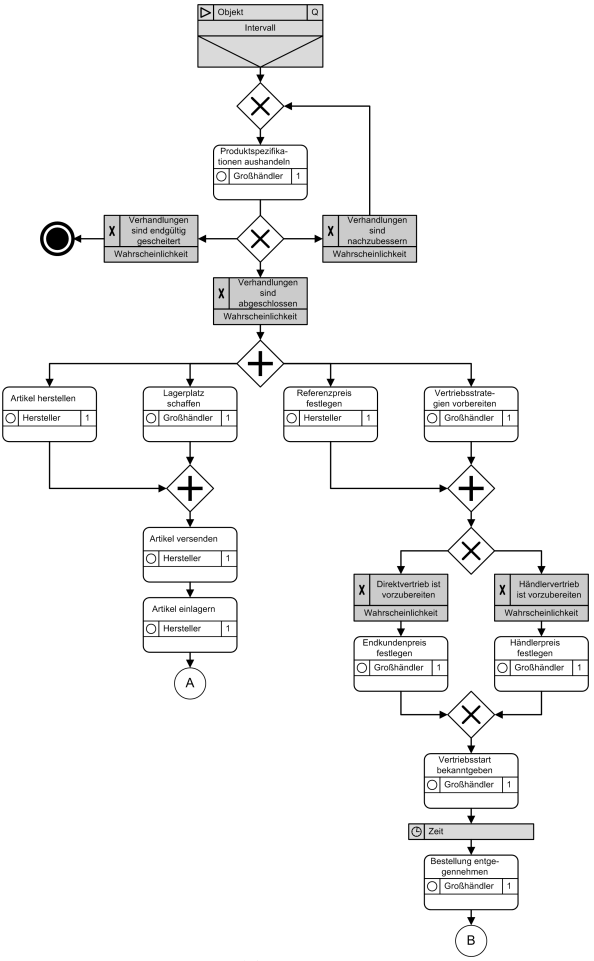


Abbildung 79: Transformationsmodell – Produkteinführung (UML) – konzeptuell – Teil 1

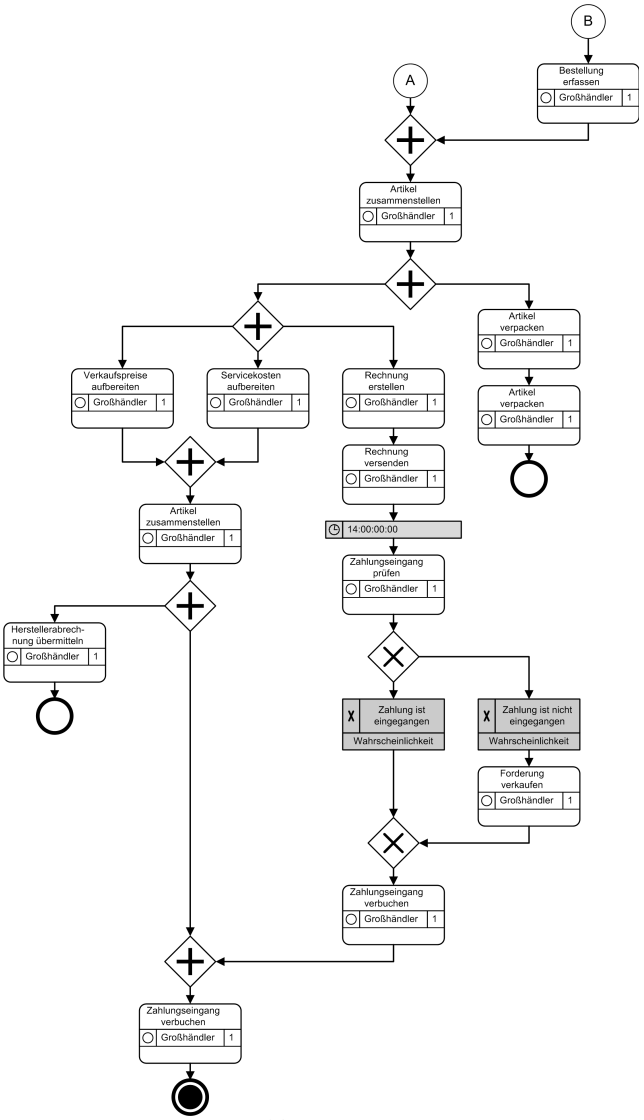
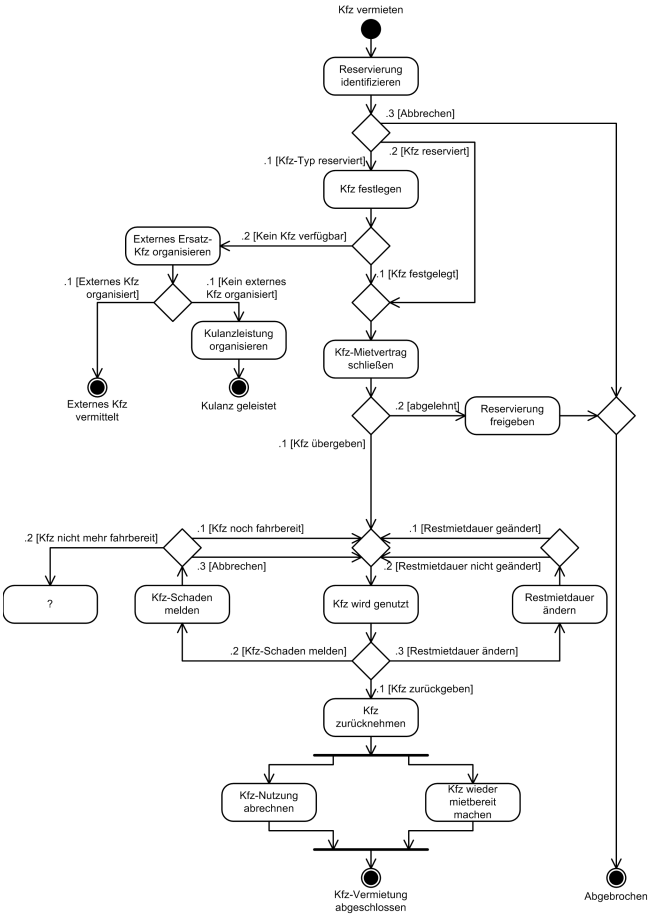


Abbildung 80: Transformationsmodell – Produkteinführung (UML) – konzeptuell – Teil 2

Kfz vermieten

Das letzte betrachtete Modell für die Evaluation der Transformationsregeln von Quellmodellen zum ProSiT Ablaufdiagramm ist der Geschäftsprozess „Kfz vermieten“. Dieses UML Aktivitätsdiagramm ist in Abbildung 81 dargestellt.



Oestereich et al. (2004, S. 112)

Abbildung 81: UML – Kfz vermieten

Anwendung der Vorbereitungsregeln

Auf das Quellmodell muss nur eine Vorbereitungsregel angewendet werden. Grund hierfür ist der Prozessablauf nach der Aktion „?“. Dieser ist nicht weiter beschrieben. Jeder Pfad muss aber über ein Endereignis verfügen. Vorbereitungsregel $adPR_{2\_5}$ erzeugt hierfür ein Flussende. Das resultierende Modell ist in Abbildung 82 dargestellt.

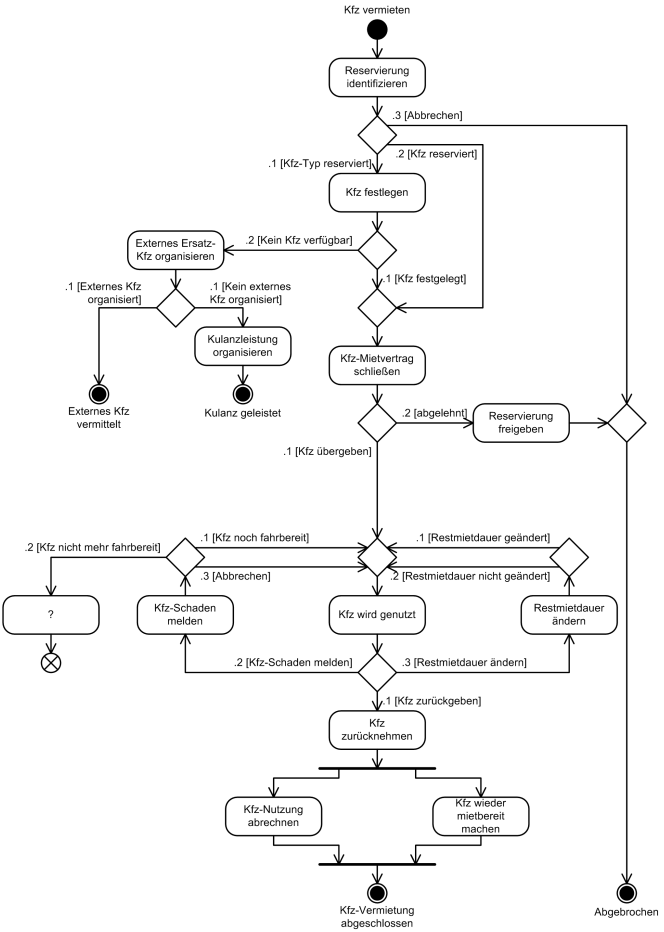
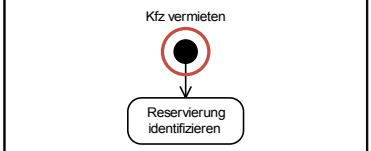
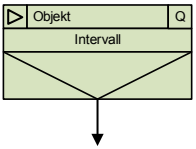
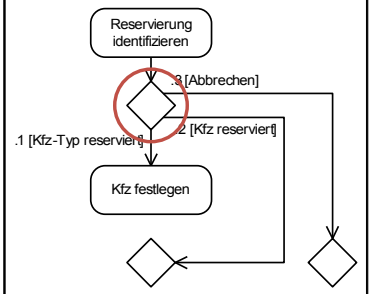
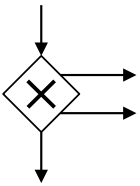
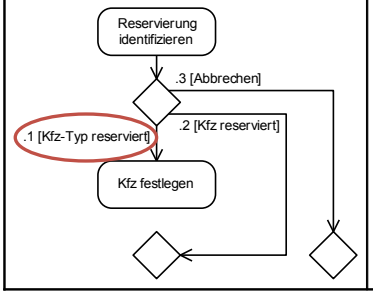
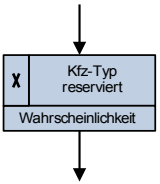


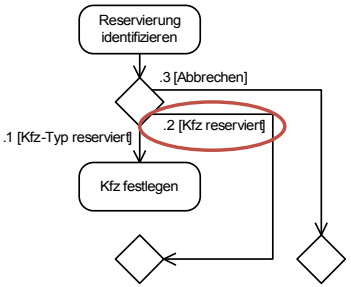
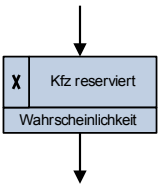
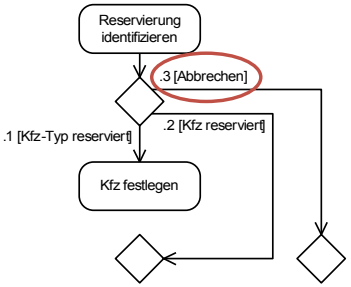
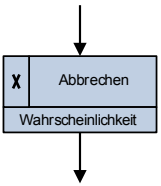
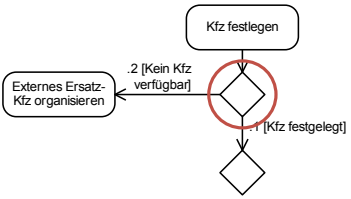
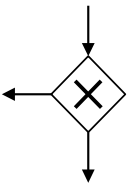
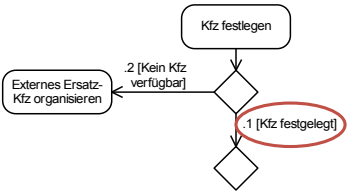
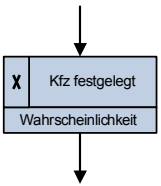
Abbildung 82: UML – Kfz vermieten – vorbereitet

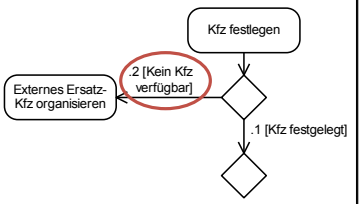
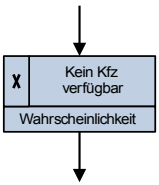
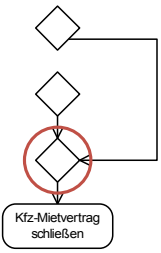
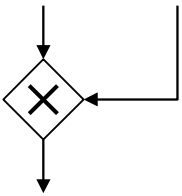
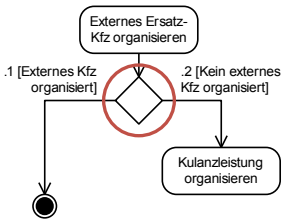
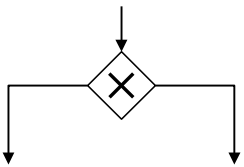
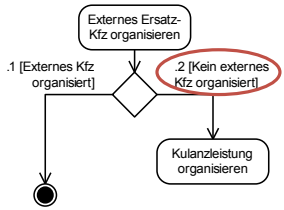
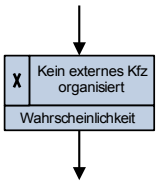
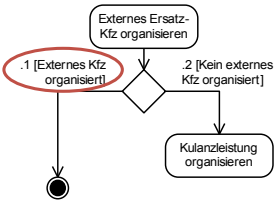
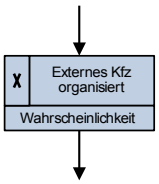
Anwendung der Transformationsregeln

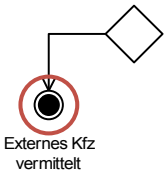
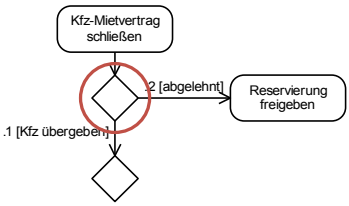
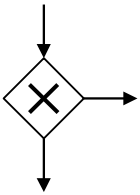
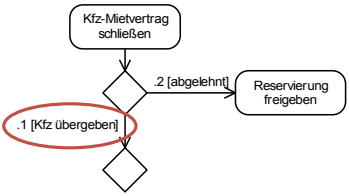
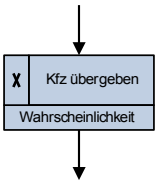
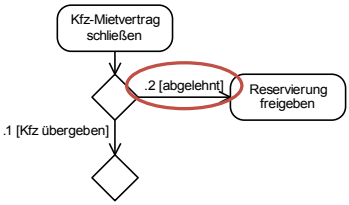
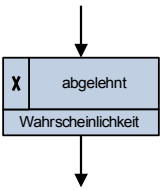
Nach dieser kleinen Änderung am Ursprungsmodell können die Transformationsregeln angewendet werden, die in Tabelle 50 aufgelistet werden.

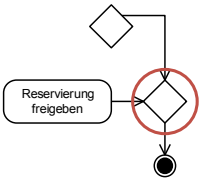
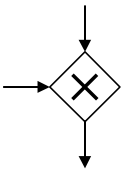
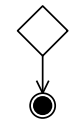
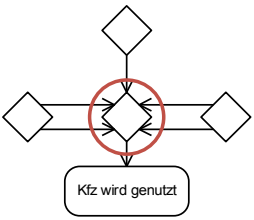
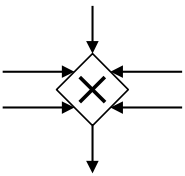
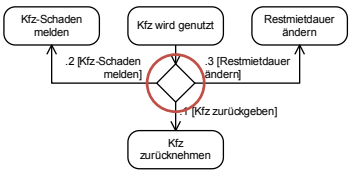
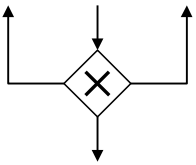
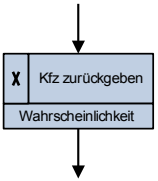
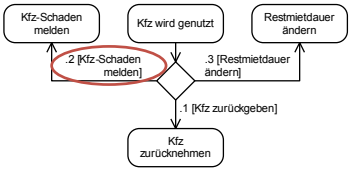
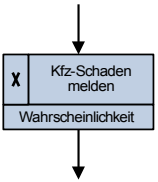
Tabelle 50: Anwendung der UML AD Transformationsregeln auf Kfz vermieten

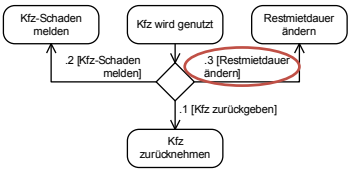
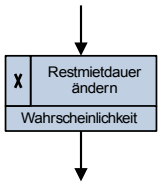
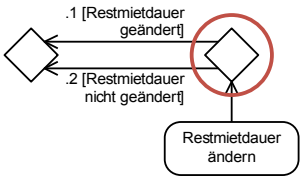
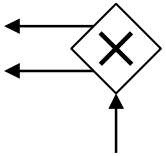
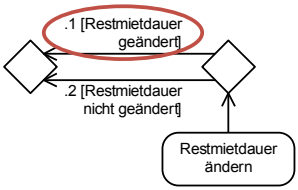
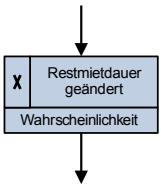
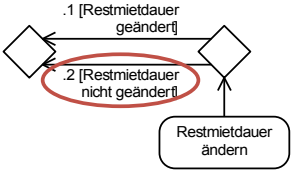
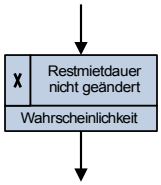
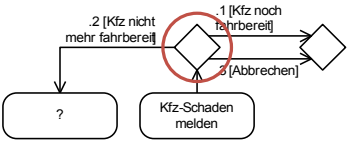
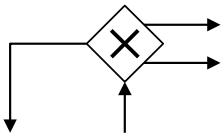
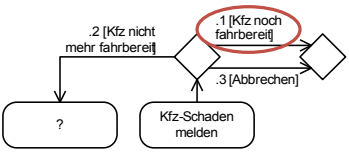
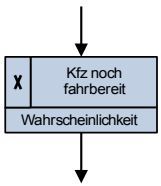
UML Element	adTR	Resultierendes ProSiT Ablaufdiagramm
	$adTR_1$	
	$adTR_{12}$	
	$adTR_{12}$	

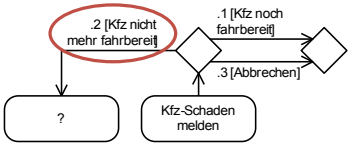
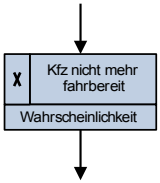
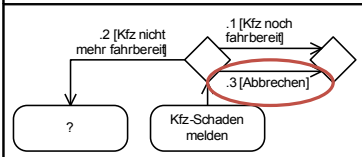
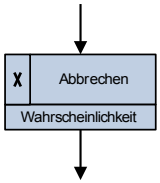
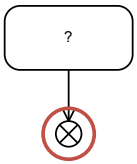
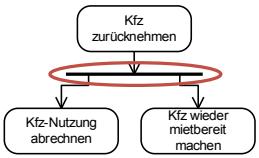
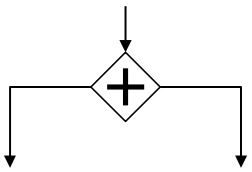
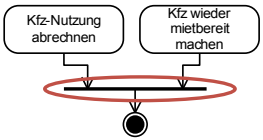
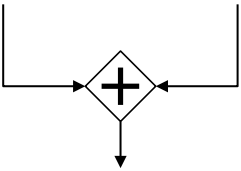
UML Element	adTR	Resultierendes ProSiT Ablaufdiagramm
	$adTR_{12}$	
	$adTR_{12}$	
	$adTR_{12}$	
	$adTR_{12}$	

UML Element	adTR	Resultierendes ProSiT Ablaufdiagramm
	$adTR_{12}$	
	$adTR_{13}$	
	$adTR_{12}$	
	$adTR_{12}$	
	$adTR_{12}$	

UML Element	adTR	Resultierendes ProSiT Ablaufdiagramm
	$adTR_5$	
	$adTR_5$	
	$adTR_{12}$	
	$adTR_{12}$	
	$adTR_{12}$	

UML Element	adTR	Resultierendes ProSiT Ablaufdiagramm
	$adTR_{13}$	
	$adTR_{5}$	
	$adTR_{13}$	
	$adTR_{12}$	
	$adTR_{12}$	
	$adTR_{12}$	

UML Element	adTR	Resultierendes ProSiT Ablaufdiagramm
	$adTR_{12}$	
	$adTR_{12}$	
	$adTR_{12}$	
	$adTR_{12}$	
	$adTR_{12}$	
	$adTR_{12}$	

UML Element	adTR	Resultierendes ProSiT Ablaufdiagramm
	$adTR_{12}$	
	$adTR_{12}$	
	$adTR_{3}$	
	$adTR_{14}$	
	$adTR_{15}$	
	$adTR_{5}$	

Konzeptuelles Transformationsmodell

Das aus diesen Transformationsregeln entstandene konzeptuelle ProSiT Ablaufdiagramm ist in Abbildung 83 dargestellt.

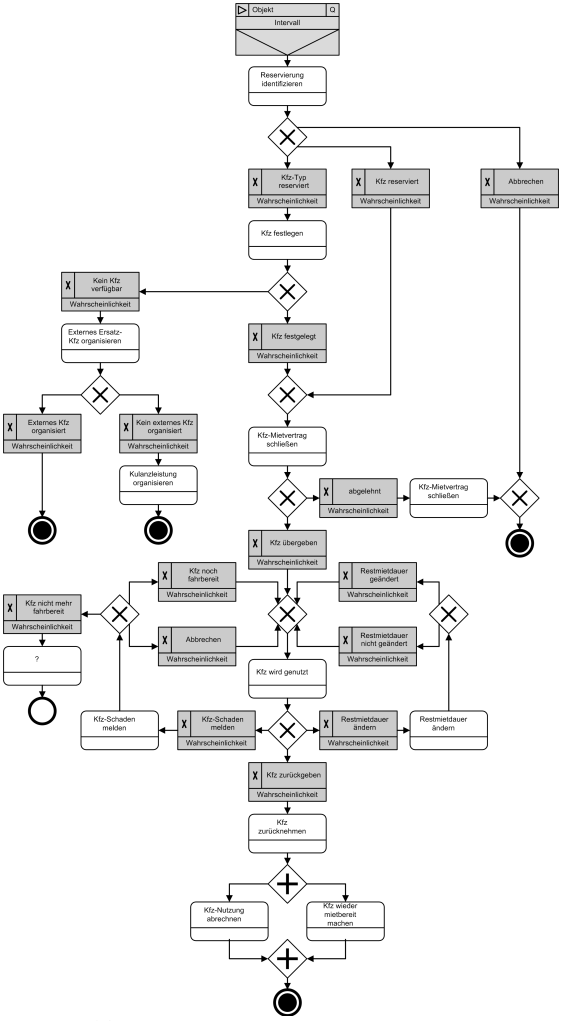


Abbildung 83: Transformationsmodell – Kfz vermieten – konzeptuell

4.3 Anwendung der Normalisierung

Die Anwendung der Normalisierungsregeln kann nicht als strenge Evaluation durchgeführt werden, da nach der Anwendung der Regeln kein richtig oder falsch bestimmt werden kann. Daher wird exemplarisch die Anwendung der Normalisierung anhand der beiden Geschäftsprozesse Chirurgischer Notfall und Voruntersuchung demonstriert. Damit an diesen Prozessen die linguistischen Normalisierungsregeln angewendet werden können, wird zunächst ein Repository definiert. Da beide Prozesse der Krankenhausdomäne zuzuordnen sind, wird für diese Domäne das Repository definiert. Gemäß dem Muster in Tabelle 18 ist die nachfolgende Klassifikationstabelle (51) für die Verben der beiden Geschäftsprozessmodelle gefüllt.

Tabelle 51: Klassifikationstabelle für die Evaluation der Normalisierung

Domäne	Verb	Prozessart	Aktivitätsart	Prozessobjekt
Krankenhaus	ablegen	KP	UA	nein
Krankenhaus	assistieren	KP	UA	ja
Krankenhaus	aufrufen	KP	UA	ja
Krankenhaus	drucken	KP	UA	nein
Krankenhaus	durchführen	KP	HA	ja
Krankenhaus	entnehmen	KP	UA	nein
Krankenhaus	entscheiden	KP	HA	ja
Krankenhaus	erfassen	KP	HA	ja
Krankenhaus	erstellen	KP	HA	ja
Krankenhaus	festlegen	KP	HA	ja
Krankenhaus	fragen	KP	HA	ja
Krankenhaus	informieren	KP	UA	nein
Krankenhaus	legen	KP	HA	ja
Krankenhaus	messen	KP	HA	ja
Krankenhaus	notieren	KP	UA	nein
Krankenhaus	prüfen	KP	HA	ja

Domäne	Verb	Prozessart	Aktivitätsart	Prozessobjekt
Krankenhaus	röntgen	KP	HA	ja
Krankenhaus	sichern	KP	HA	ja
Krankenhaus	übernehmen	KP	UA	ja
Krankenhaus	überwachen	KP	UA	ja
Krankenhaus	umlagern	KP	UA	ja
Krankenhaus	vorbereiten	KP	UA	nein
Krankenhaus	zurückgeben	KP	UA	ja

Neben der Klassifikationstabelle wird darüber hinaus eine einfache Synonymtabelle definiert, welche in Tabelle 52 aufgeführt wird.

Tabelle 52: Synonymtabelle für die Evaluation der Normalisierung

Domäne	Synonym	Verb
Krankenhaus	bestimmen	festlegen
Krankenhaus	benachrichtigen	informieren
Krankenhaus	vermessen	messen

Basierend auf der Klassifikationstabelle (51) wird als letzte Tabelle (53) im Rahmen der Tätigkeitssicht die Vorschlagstabelle für die Erfassungsmethode aufgeführt.

Tabelle 53: Erfassungsmethodenvorschlag für die Evaluation der Normalisierung

Domäne	Verb	Prozessart	Aktivitätsart	Methode
Krankenhaus	ablegen	KP	UA	schätzen
Krankenhaus	assistieren	KP	UA	befragen
Krankenhaus	aufrufen	KP	UA	schätzen
Krankenhaus	drucken	KP	UA	schätzen
Krankenhaus	durchführen	KP	HA	messen
Krankenhaus	entnehmen	KP	UA	schätzen
Krankenhaus	entscheiden	KP	HA	befragen

Domäne	Verb	Prozessart	Aktivitätsart	Methode
Krankenhaus	erfassen	KP	HA	messen
Krankenhaus	erstellen	KP	HA	messen
Krankenhaus	festlegen	KP	HA	messen
Krankenhaus	fragen	KP	HA	befragen
Krankenhaus	informieren	KP	UA	schätzen
Krankenhaus	legen	KP	UA	messen
Krankenhaus	messen	KP	HA	befragen
Krankenhaus	notieren	KP	UA	schätzen
Krankenhaus	prüfen	KP	HA	befragen
Krankenhaus	röntgen	KP	HA	messen
Krankenhaus	sichern	KP	HA	messen
Krankenhaus	übernehmen	KP	UA	befragen
Krankenhaus	überwachen	KP	UA	befragen
Krankenhaus	umlagern	KP	UA	befragen
Krankenhaus	vorbereiten	KP	UA	befragen
Krankenhaus	zurückgeben	KP	UA	schätzen

Chirurgischer Notfall

Am konzeptionellen Modell des chirurgischen Notfalls werden als erstes die manuellen Normalisierungen mNR_1, mNR_2 und mNR_3 angewendet. Hiermit wird die Domäne bestimmt, der Prozesstyp und das Prozessobjekt für das ProSiT Ablaufdiagramm. Die Domäne des Geschäftsprozesses ist das Krankenhaus (mNR_1). Der chirurgische Notfall im Rahmen der Notaufnahme stellt einen Kernprozess dar (mNR_2) Das Prozessobjekt für diesen Prozess ist der Patient (mNR_3).

Nach der Durchführung dieser drei Normalisierungsschritte können die automatischen Normalisierungsregeln aNR_{13} und aNR_{14} angewendet werden. Beispielhaft ist die Anwendung beider Regeln in Abbildung 84 dargestellt.

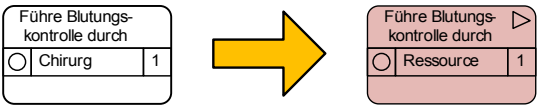


Abbildung 84: Beispielhafte Anwendung der Normalisierungsregeln aNR_{13} und aNR_{14}

Die Anwendung dieser beiden Normalisierungsregeln auf die Aktivitäten des chirurgischen Notfalls führt zur Zuordnung dargestellt in Tabelle 54.

Tabelle 54: Anwendung der Regeln aNR_{13} und aNR_{14} auf den chirurgischen Notfall

Aktivität	Verb	Aktivitätsart	Prozessobjekt
Informiere Notfallpersonal	informieren	UA	nein
Übernehme Patient vom Rettungsdienst	übernehmen	UA	ja
Erstelle Notfallanamnese	erstellen	HA	ja
Lagere Patienten um	umlagern	UA	ja
Führe Ultraschall durch	durchführen	HA	ja
Führe Blutungskontrolle durch	durchführen	HA	ja
Führe Notfallendoskopie durch	durchführen	HA	ja
Sichere Atmung	sichern	HA	ja
Sichere Kreislauf	sichern	HA	ja
Überwache Vitalfunktionen	überwachen	UA	ja
Lege Zugänge	legen	HA	ja
Assistiere der Erstversorgung	assistieren	UA	ja
Röntege Thorax	röntgen	HA	ja
Röntege Abdomenübersicht	röntgen	HA	ja
Führe orientierende Untersuchung durch	durchführen	HA	ja
Bereite Blutuntersuchung vor	vorbereiten	UA	nein
Führe Blutuntersuchung durch	durchführen	HA	ja
Entscheide Therapie	entscheiden	HA	ja
Prüfe, ob Blasenkatheter erforderlich	prüfen	HA	ja

Aktivität	Verb	Aktivitätsart	Prozessobjekt
Lege Blasenkatheter	legen	HA	ja
Bestimme weitere Untersuchungen	bestimmen	HA	ja
Erfasse EKG	erfassen	HA	ja
Erfasse Blutdruck	erfassen	HA	ja
Führe Endoskopie durch	durchführen	HA	ja
Prüfen, ob Bronchoskopie erforderlich	prüfen	HA	ja
Führe Bronchoskopie durch	durchführen	HA	ja
Erstelle Bilanzierung	erstellen	HA	ja

Die letzte Aktivität „Halte interdisziplinäres Konzil ab“ ist nicht in Tabelle 54 enthalten. Grund hierfür ist das Verb „abhalten“. Dieses Verb ist nicht im Repository (Tabelle 51) definiert, weshalb die Normalisierungsregel aNR_{13} und aNR_{14} bei der Aktivität nicht aktiv werden. Bei der Aktivität greifen jedoch die semiautomatischen Normalisierungsregeln sNR_6 und sNR_7, die – analog zu den automatischen Normalisierungsregeln – den Modellierer auffordern, die Aktivität zu klassifizieren. Die Aktivität „Halte interdisziplinäres Konzil ab“ wird als Hauptaktivität klassifiziert und hinsichtlich der Durchführung nicht am Prozessobjekt. Da im Rahmen des Konzils der Gesundheitsstand besprochen wird und weitere Behandlungsschritte festgelegt werden, wird diese als Hauptaktivität aufgefasst. Während des Konzils ist der Patient aber nicht anwesend, weshalb die Ausführung nicht am Prozessobjekt erfolgt. Durch die Anwendung dieser vier Normalisierungsregeln, ist das gesamte ProSiT Ablaufdiagramm entsprechend klassifiziert.

Bevor die semiautomatische Normalisierungsregel sNR_2 aktiv wird, welche nach sequenziell verwendeten Ressourcen prüft, werden die manuellen Normalisierungsschritte mNR_4 und mNR_5 durchgeführt. Im Rahmen der manuellen Normalisierung mNR_4 wird als stationäre Ressource ein Raum für die „Notaufnahme“ bei fast jeder Aktivität hinzugefügt sowie bei den Aktivitäten mit dem Röntgen der

Röntgenraum. Im Rahmen der Ressourcensicht werden mit Ausnahme der „Notaufnahme“ alle Ressourcen als Menschen deklariert, womit diese auch im ProSiT Ablaufdiagramm entsprechend gekennzeichnet sind. Ebenfalls werden manuell Ressourcenbindungen und -freigaben in das Modell eingefügt, was die manuelle Normalisierung mNR_5 vorsieht. Das Zwischenergebnis nach der Anwendung der bisherigen Normalisierungsregeln ist in den Abbildungen 85, 86 und 87 dargestellt.

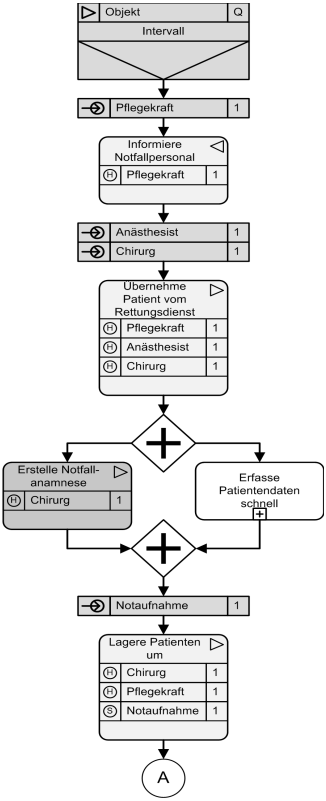


Abbildung 85: Teilnormalisiertes konzeptionelles ProSiT Ablaufdiagramm – Chirurgischer Notfall – Teil 1

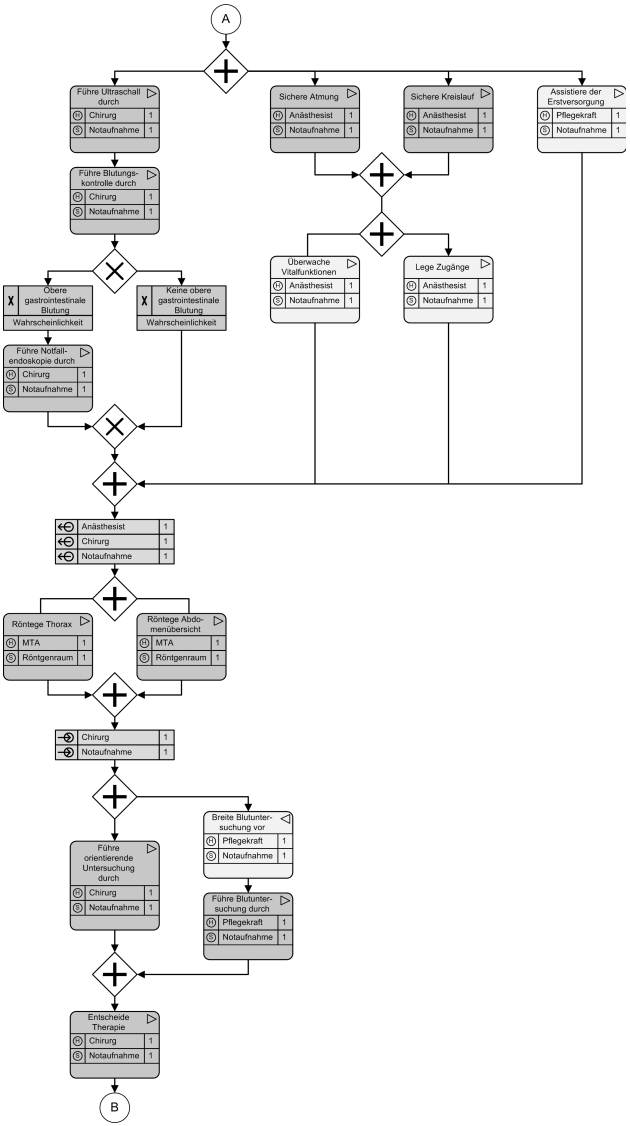


Abbildung 86: Teilnormalisiertes konzeptionelles ProSiT Ablaufdiagramm –Chirurgischer Notfall – Teil 2

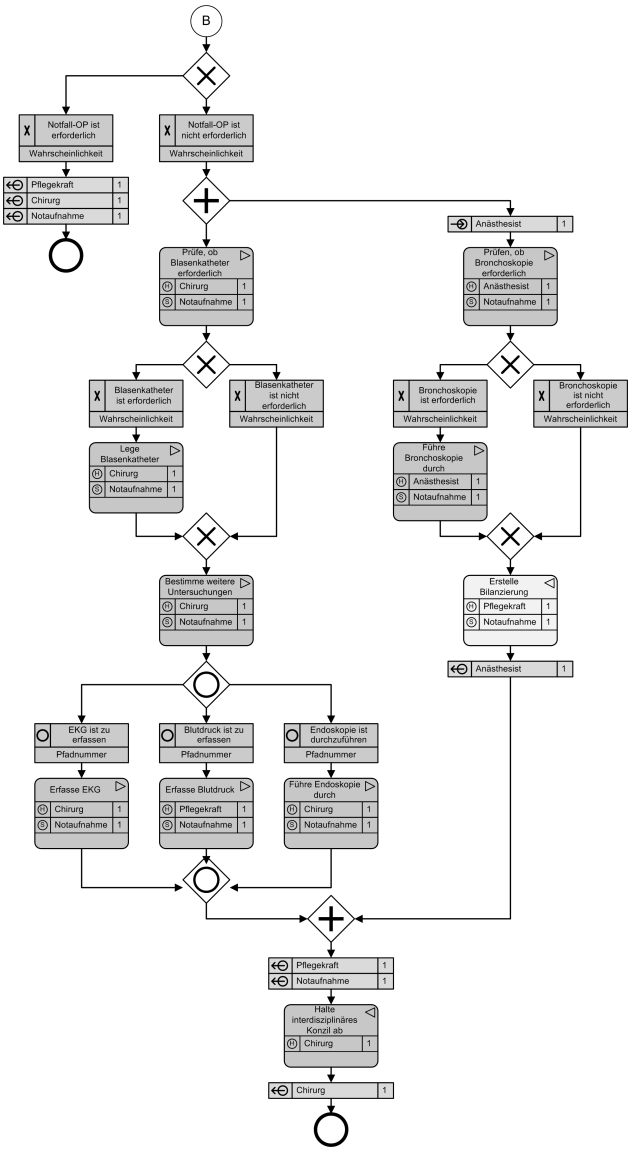


Abbildung 87: Teilnormalisiertes konzeptionelles ProSiT Ablaufdiagramm –Chirurgischer Notfall – Teil 3

Nachdem die Aktivitäten klassifiziert sind und alle relevanten Ressourcen dem ProSiT Ablaufdiagramm hinzugefügt wurden, können darauf aufbauende Regeln angewendet werden. So schlägt die semiautomatische Normalisierungsregel sNR_3 zwei Mal beim Chirurgischen Notfall an (Abbildung 88).

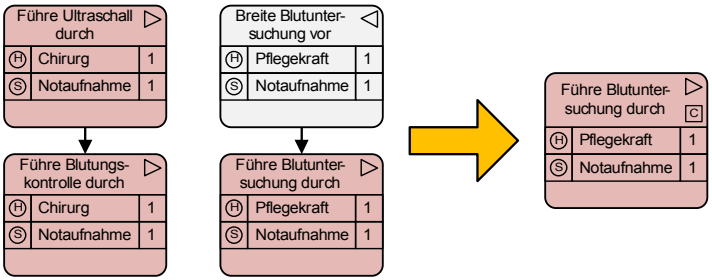


Abbildung 88: Anwendung der Normalisierungsregel sNR3 auf den Chirurgischen Notfall

Auf die linken zwei Aktivitäten wird die Normalisierungsregel nicht angewendet, da diese zwei verschiedene Tätigkeiten darstellen. Die rechten Aktivitäten, „Bereite Blutuntersuchung vor“ und „Führe Blutuntersuchung durch“ hingegen beziehen sich auf die gleiche durchzuführende Tätigkeit, die Blutuntersuchung. Beide Aktivitäten werden daher in der Aktivität „Führe Blutuntersuchung durch“ als kombinierte Aktivität zusammengefasst.

Bezogen auf die Bearbeitungszeit wird bei den automatischen Normalisierungsregeln aNR_{16} tätig. Die Aktivität „Überwache Vitalfunktionen“ wird zeitabhängig definiert. Eine Ressourcenbindung und -freigabe wird nicht in das Modell eingefügt, da die Ressourcen bereits gebunden sind. Das gleiche Ergebnis kann die semiautomatische Normalisierungsregel sNR_8 hervorbringen. Diese fragt den Modellierer bei zwei Aktivitäten, ob gleich verfahren werden soll, bei der Aktivität „Lege Zugänge“ und bei „Assistiere der Erstversorgung“. Angewendet wird die Regel aber nur bei der zweit genannten Aktivität.

Mit Ausnahme der Zwei als zeitabhängig klassifizierten Aktivitäten müssen alle anderen Aktivitäten mit einer Bearbeitungszeit versehen

werden. Da keine realitätsgetreuen Zeiten im Rahmen der Arbeit verwendet werden, wird für die Zeitfestlegung eine Zuordnung anhand der Zeiterfassungsmethode vollzogen. Die aus der Erfassungsmethode resultierende Bearbeitungszeit ist in Tabelle 55 aufgeführt.

Tabelle 55: Bearbeitungszeiten für den chirurgischen Notfall

Erfassungsmethode	Bearbeitungszeit
messen	Normal: 4:00, 0:45
befragen	Dreieck: 1:00, 2:00, 4:00
schätzen	01:00:00

Durch die Verwendung der Erfassungsmethode kann für das Modell angegeben werden, dass die semiautomatische Normalisierungsregel sNR_{13} indirekt angewendet wird. Das Überführen der Zeiten nimmt jedoch die manuelle Normalisierung mNR_7 für sich in Anspruch. Beide Regeln für die Zeitdefinition von Aktivitäten werden daher im Rahmen des chirurgischen Notfalls berücksichtigt.

Hinsichtlich nicht gepflegter Attribute stehen, neben der Quelle, noch die Gateways aus. Zu pflegen sind zum einen das verzweigende exklusive sowie das inklusive Gateway.

Auf die beiden inklusiven Gateways wird zunächst die Vorbereitungsregel sNR_9 angewendet. Da die Notversorgung erst abgeschlossen werden kann, wenn alle Aktivitäten zwischen den inklusiven Gateways abgeschlossen sind, werden beide als „wait-for-all“ klassifiziert. An die Anwendung diese Vorbereitungsregel kann die Anwendung der manuellen Normalisierung mNR_{11} angeschlossen werden. Im Rahmen dieser sind die inklusiven Schlüssel mit Pfadnummern zu belegen und es ist eine Entscheidungstabelle für die inklusive Entscheidung aufzubauen. Die inklusiven Schlüssel sind mit einer aufsteigenden Nummer nummeriert, beginnend ab „A1“ und die Entscheidungstabelle ist in Tabelle 56, beispielhaft mit Wahrscheinlichkeiten gefüllt, dargestellt.

Tabelle 56: Inklusive Entscheidungstabelle des chirurgischen Notfalls

Wahrscheinlichkeitstabelle A	
Pfad	**Wahrscheinlichkeit**
2	10,0%
12	35,0%
23	35,0%
123	20,0%

Für die exklusiven Schlüssel der verzweigenden Gateways muss noch die manuelle Normalisierung mNR_{10} angewendet werden. Im Rahmen der Evaluation wird festgelegt, dass ein positives Ergebnis der Prüfung viermal häufiger eintritt, als das Negative. Eine Ausnahme davon ist die Entscheidung über die Notfall-OP, diese wird entgegengesetzt definiert.

Als letztes Element muss die Quelle des chirurgischen Notfalls definiert werden. Diese Definition ist Gegenstand der manuellen Normalisierung mNR_9. Als Intervall wird eine Normalverteilung mit Mittelwert von 7 Minuten und einer Standardabweichung von 1 Minute angegeben sowie für die Quantität 1 Patient, gemäß des Prozessobjekts. Das aus der gesamten Normalisierung resultierende konsistente ProSiT Ablaufdiagramm ist in den Abbildungen 89 und 90 dargestellt.

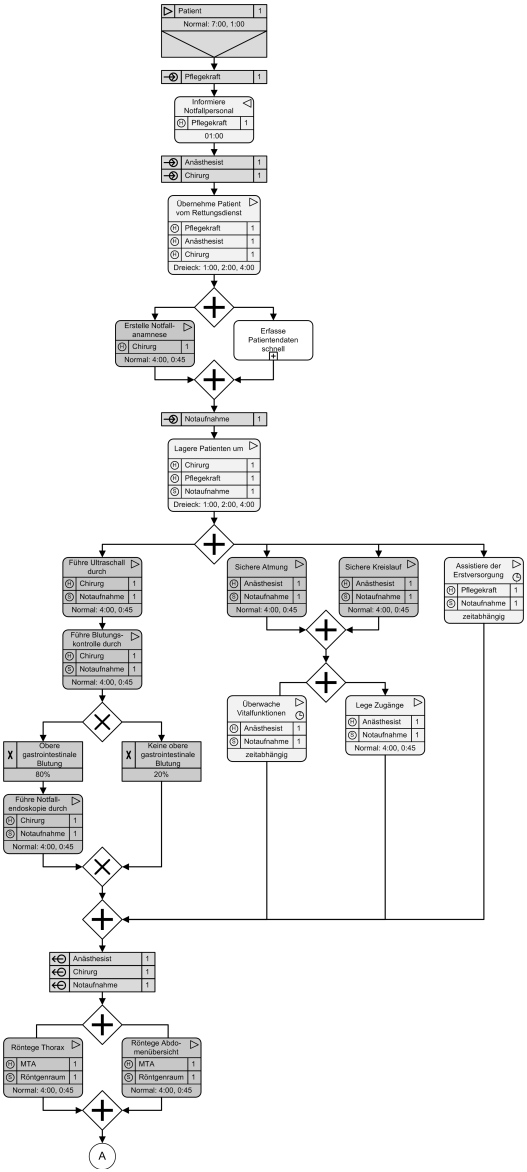


Abbildung 89: Konsistentes ProSiT Ablaufdiagramm des chirurgischen Notfalls – Teil 1

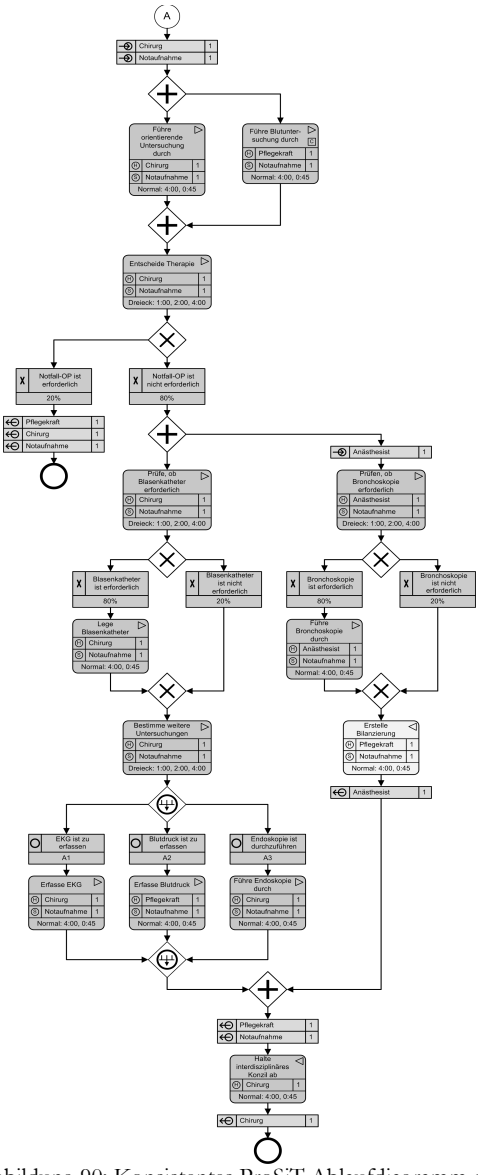


Abbildung 90: Konsistentes ProSiT Ablaufdiagramm des chirurgischen Notfalls – Teil 2

Voruntersuchung

Die Normalisierung der Voruntersuchung verläuft nach dem gleichen Schema wie beim chirurgischen Notfall. So wird die Domäne (mNR_1) des Ablaufdiagramms mit Krankenhaus festgelegt, das Modell als Kernprozess (mNR_2) klassifiziert und der Patient als Prozessobjekt (mNR_3) bestimmt. Mit diesen Definitionen kann die Klassifikation der Aktivitäten mit den automatischen Normalisierungsregeln aNR_{13} und aNR_{14} erfolgen. Eine Übersicht dieser Klassifikation ist in Tabelle 57 aufgeführt.

Tabelle 57: Anwendung der Normalisierungsregeln aNR_{13} und aNR_{14} auf die Voruntersuchung

Aktivität	Verb	Aktivitätsart	Prozessobjekt
Ambulanzakte zur Untersuchung entnehmen	entnehmen	UA	nein
Patient aufrufen	aufrufen	UA	ja
Patient die Chipkarte zurückgeben	zurückgeben	UA	ja
Status des Patienten prüfen	prüfen	HA	ja
Mobilisierbarkeit des Patienten prüfen	prüfen	HA	ja
Patient in das Untersuchungszimmer bitten	bitten		ja
Augendruck messen	messen	HA	ja
Untersuchungsergebnis drucken	drucken	UA	nein
Refraktion bestimmen	bestimmen	HA	ja
Sehschärfe mit Sehprobentafel messen	messen	HA	ja
Patient nach einer vorhandenen Sehhilfe fragen	fragen	HA	ja
Sehhilfe vermessen	vermessen	HA	ja
Sehschärfe mit Sehzeichenprojektor messen	messen	HA	ja

Aktivität	Verb	Aktivitätsart	Prozessobjekt
Untersuchungsergebnis notieren	notieren	UA	nein
Anamnese durchführen	durchführen	HA	ja
Befragungsergebnisse notieren	notieren	UA	nein
Nächste Untersuchungsschritte prüfen	prüfen	HA	ja
Patient vor Behandlungsraum platzieren	platzieren		ja
Ambulanzakte in Behandlungsraum ablegen	ablagen	UA	nein
Patient vor Untersuchungsraum platzieren	platzieren		ja
Ambulanzakte in Untersuchungsraum ablegen	ablegen	UA	nein
Untersuchenden Mitarbeiter benachrichtigen	benachrichtigen	UA	nein
Patient vor Sehschule platzieren	platzieren		ja
Ambulanzakte in Sehschule ablegen	ablegen	UA	nein

Fünf Aktivitäten können nicht vollständig über das Repository klassifiziert werden, da die Verben „bitten“, „kleben“ und „platzieren“ nicht enthalten sind. Da in der Tätigkeitsbeschreibung aber der Patient – das Prozessobjekt – enthalten ist, werden vier der fünf Aktivitäten hinsichtlich des Prozessobjekts klassifiziert. Die verbleibende Aktivität muss aber noch entsprechend klassifiziert werden, beziehungsweise für alle fünf muss noch die Aktivitätsart bestimmt werden. Eine Übersicht der manuellen Klassifizierung ist in Tabelle 58 aufgeführt. Neben diesen fünf Aktivitäten wird bei den drei markierten Aktivitäten die Aktivitätsart von Hauptaktivität in Unterstützungsaktivität manuell umgeändert.

Tabelle 58: Manuelle Klassifizierung bei der Voruntersuchung nach sNR_6 und sNR_7

Aktivität	Verb	Aktivitätsart	Prozessobjekt
Status des Patienten prüfen	prüfen	**UA**	ja
Mobilisierbarkeit des Patienten prüfen	prüfen	**UA**	ja
Patient in das Untersuchungszimmer bitten	bitten	UA	ja
Patient nach einer vorhandenen Sehhilfe fragen	fragen	**UA**	ja
Untersuchungsergebnisse in Ambulanzakte kleben	kleben	UA	nein
Patient vor Behandlungsraum platzieren	platzieren	UA	ja
Patient vor Untersuchungsraum platzieren	platzieren	UA	ja
Patient vor Sehschule platzieren	platzieren	UA	ja

Das Ergebnis dieser Klassifizierung ist in den Abbildungen 91 und 92 als Zwischenstand dargestellt. Im direkten Vergleich zum teilnormalisieren konzeptionellen ProSiT Ablaufdiagramm des chirurgischen Notfalls fällt auf, dass die Voruntersuchung im Verhältnis viel mehr Unterstützungsaktivitäten aufweist. So weist der chirurgische Notfall 20 Haupt- und 8 Unterstützungsaktivitäten und somit einen Anteil von grob 30% an Unterstützungsaktivitäten. Bei der Voruntersuchung hingegen kann grob von einem umgekehrten Verhältnis gesprochen werden, da diese 8 Haupt- und 25 Unterstützungsaktivitäten aufweist. Das grobe Verhältnis liegt daher bei etwa 80%.

Abgeleitet aus diesen Verhältnissen sei die These aufgestellt, dass ein hoher Anteil an Unterstützungsaktivitäten an der Gesamtanzahl an Aktivitäten auf einen zu detailliert modellierten Geschäftsprozess hinweist. Verallgemeinert könnte aus dieser These auch abgeleitet werden, dass ein Geschäftsprozessmodell mit hohem Anteil an Unterstützungsaktivitäten auf eine hohe Granularität hinweist. Diese These wird aber im Rahmen der Arbeit nicht weiter untersucht,

sondern wird als eine offene Forschungsfrage aufgefasst, die weitere Untersuchungen bedarf.

Im Ablaufdiagramm ist der Funktions-/Medizinisch-/Technische Dienst bereits als Mensch klassifiziert. Dies erfolgte über die Ressourcensicht, in dem bei dieser der Dienst entsprechend klassifiziert wurde.

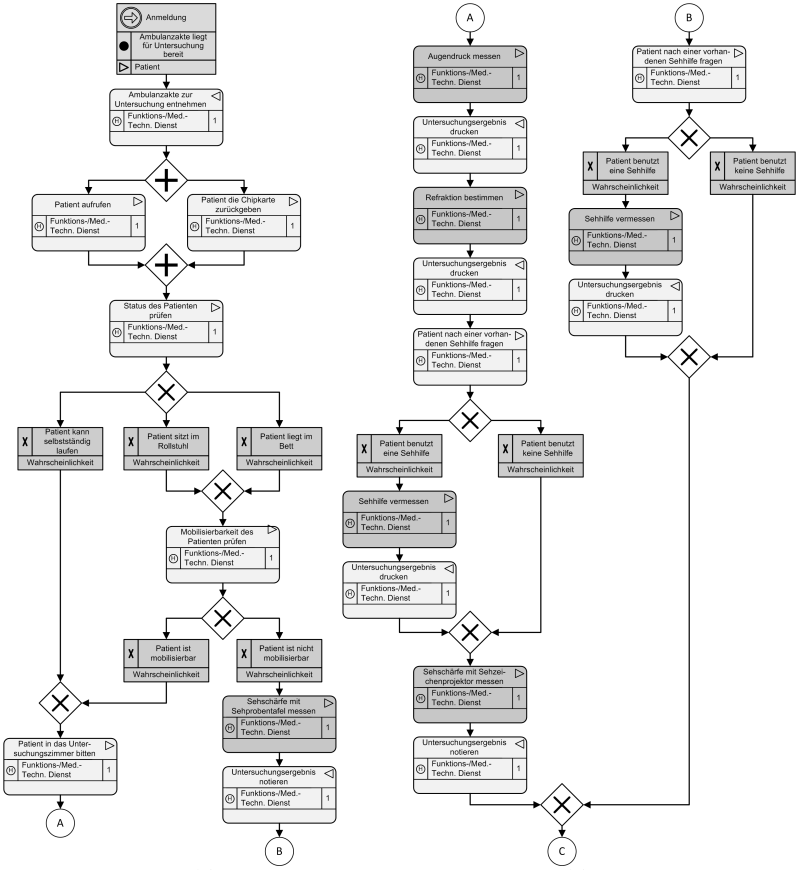


Abbildung 91: Teilnormalisiertes konzeptionelles ProSiT Ablaufdiagramm – Voruntersuchung – Teil 1

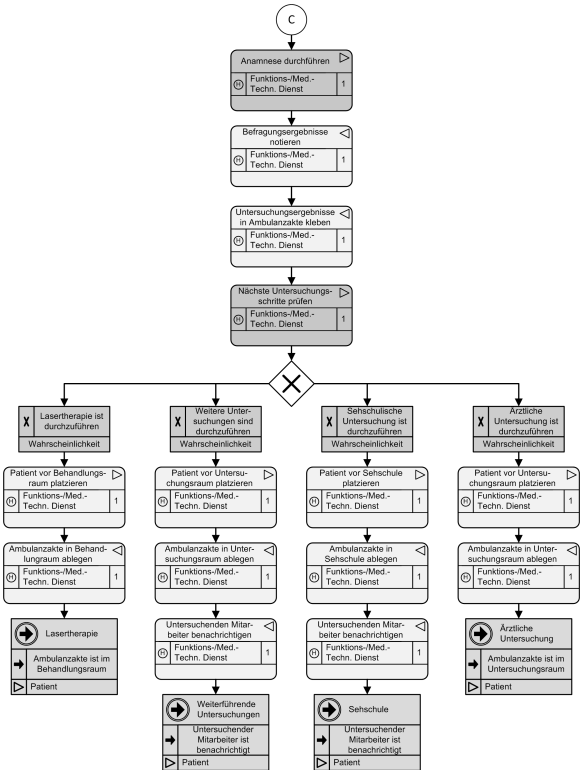


Abbildung 92: Teilnormalisiertes konzeptionelles ProSiT Ablaufdiagramm – Voruntersuchung – Teil 2

Hinsichtlich der Ressourcen wird zweimal die automatische Normalisierungsregel aNR_{15} fündig. Beide Aktivitäten, „Sehschärfe mit Sehprobentafel messen“ und „Sehschärfe mit Sehzeichenprojektor messen“ weisen aber eine Ressource auf, die nicht simulationsrelevant ist. Während die automatische Normalisierungsregel aNR_{15} diese Ressourcen den Aktivitäten zuweist, werden diese durch die manuelle Normalisierung mNR_4 wieder entfernt werden. Durch die Anwendung beider Normalisierungen ändert sich aber die Tätigkeit der Aktivitäten jeweils in „Sehschärfe messen“.

Der Prozessablauf bedingt im Rahmen der manuellen Normalisierung mNR_4 eine weitere stationäre Ressource, den Raum für die Voruntersuchung. Dieser wird bei allen Aktivitäten im linken Prozesspfad hinzugefügt, der mit der Aktivität „Patient in das Untersuchungszimmer bitten“ beginnt. Nach diesem Vorgehen würde der Raum für die Voruntersuchung bis zur Aktivität „Anamnese durchführen“ eingetragen. Für die mit der Anamnese beginnenden vier sequenziellen Aktivitäten (Abbildung 93) liegt jetzt aber ein Problem vor, dass als Anamnese Problem bezeichnet sei.

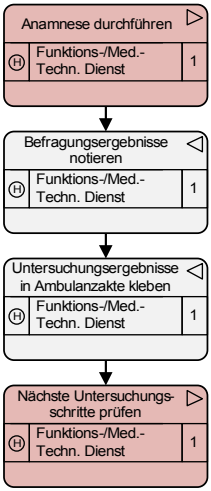


Abbildung 93: Anamnese Problem bei der Voruntersuchung

Das Anamnese Problem ist dadurch gekennzeichnet, dass für das Simulationsmodell die stationäre Ressource „Voruntersuchung“ verwendet wird, während beim Geschäftsprozessmodell lediglich Stellen verwendet werden. Aus Sicht des Geschäftsprozessmodells ist der Ablauf korrekt, da nach der vorangehenden Zweiteilung die vier Aktivitäten immer ausgeführt werden. Aus Sicht des Simulationsmodells wird aber der linke vorangehende Prozesspfad in einem Raum durchgeführt, während der rechte Prozesspfad den Raum nicht benötigt. Entsprechend wird der Ablauf in Abbildung 93 einmal in der Voruntersuchung ausgeführt und einmal nicht. Damit dies berück-

sichtigt werden kann, muss der Ablauf zwei Mal im ProSiT Ablaufdiagramm vorkommen. Der durch diese manuelle Normalisierung entstehende Prozessabschnitt ist in Abbildung 94 dargestellt. Die Lösung des Anamnese Problems kann der manuellen Normalisierung mNR_6 zugeordnet werden. Die Erkennung des Anamnese Problems und dessen Lösung ist somit abhängig vom Modellierer.

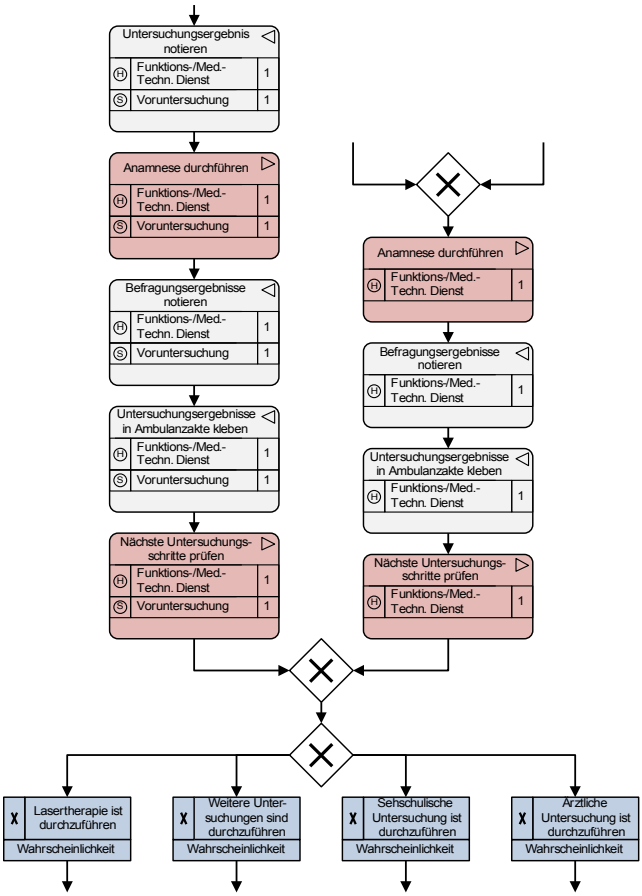


Abbildung 94: Lösung des Anamnese Problems bei der Voruntersuchung

Hinsichtlich der Ressourcen kann für den Funktions-/Medizinisch-/Technischen Dienst die manuelle Normalisierung mNR_5 angewendet

werden. So ist der Dienst von der ersten bis zur letzten Aktivität an den Prozess gebunden. Hierfür wird nach dem Zielpunkt eine Ressourcenbindung und vor den vier Sprungpunkten eine Ressourcenfreigabe für den Dienst eingefügt. Für die zweite verwendete Ressource, der Raum für die Voruntersuchung, könnte auch diese manuelle Normalisierung angewendet werden. Da sich die Ressource aber nur in einem Zweig befindet, wird die semiautomatische Normalisierungsregel sNR_2 fündig. Beim Bestätigen wird eine Ressourcenbindung vor der Aktivität „Patient in das Untersuchungszimmer bitten“ eingefügt und nach der linken Aktivität „Nächste Untersuchungsschritte prüfen“ eine Ressourcenfreigabe für die „Voruntersuchung“.

Die semiautomatische Normalisierungsregel sNR_1 wird in ProSiT Ablaufdiagramm der Voruntersuchung mehrfach aktiv. Die Regel sucht nach zwei sequenziellen Aktivitäten, bei denen die Erste eine Hauptaktivität ist, die am Prozessobjekt ausgeführt wird und die Zweite eine Unterstützungsaktivität, die nicht am Prozessobjekt ausgeführt wird, ist. Beispielhaft ist eines dieser Funde in Abbildung 95 aufgeführt. Durch die detaillierte Modellierung des Geschäftsprozesses wird zum einen der Patient untersucht und anschließend werden die Ergebnisse der Untersuchung notiert oder anderweitig aufbewahrt. Diese Aktivitäten werden jeweils zu einer kombinierten Aktivität zusammengefasst.

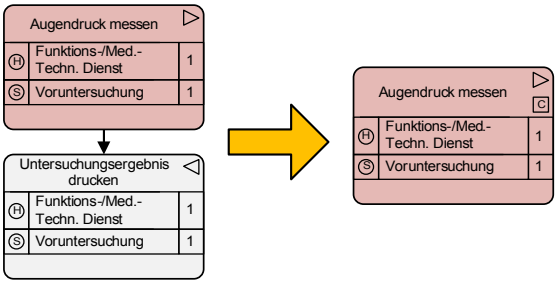


Abbildung 95: Anwendung der Regel sNR1 bei der Voruntersuchung

Insgesamt wird die semiautomatisch Normalisierungsregel sNR_1 bei der Voruntersuchung 8 mal angewendet. Da als Ressourcen nur der Funktions-/Medizinisch-/Technische Dienst sowie der Raum der Voruntersuchung verwendet werden, kann prinzipiell auch die semiautomatische Normalisierungsregel sNR_3 verwendet werden. Diese würde aber aufgrund der Anzahl an sequenziellen Aktivitäten, das Modell auf wenige Aktivitäten zusammenkürzen und auch Aktivitäten zusammenfassen, die separate Aktivitäten darstellen, weshalb die Regel nicht angewendet wird. Ebenfalls nicht angewendet wird die Normalisierungsregel sNR_8. Diese würde die zwei parallelen Aktivitäten „Patient aufrufen“ und „Patient die Chipkarte zurückgeben“ in zeitabhängigen Aktivitäten umwandeln, wodurch alle parallelen Aktivitäten zeitabhängig wären, womit aber keine Zeitabhängigkeit mehr vorliegen würde. Dieser Fall verdeutlicht jedoch, dass die Normalisierungsregeln weiter untersucht werden müssen und entsprechend umzuformulieren ist, damit dieser Fall stets ausgeschlossen wird.

Um die Normalisierung abzuschließen, sind die Bestimmung der Bearbeitungszeiten der Aktivitäten sowie der Wahrscheinlichkeiten der verzweigenden exklusiven Gateways (mNR_{10}) beziehungsweise dessen exklusive Schlüssel notwendig. Bei beiden Schritten wird das gleiche Verfahren verwendet, welches auch beim chirurgischen Notfall Anwendung gefunden hat. Bei den exklusiven Entscheidungen mit zwei Nachfolgern werden für den positiven Fall 80% und für den negativen Fall die restlichen 20% verwendet. Bei der Prüfung nach dem Status des Patienten wird angenommen, dass lediglich 10% jeweils im Rollstuhl sitzen oder im Bett liegen. Bei der letzten Prüfung mit den vier Nachfolgern wird jede Möglichkeit mit 25% angegeben.

Hinsichtlich der Bearbeitungszeit werden die Aktivitäten wieder mit dem Erfassungsmethodenvorschlag in der Tabelle 53 auf Seite 375 abgeglichen (sNR_{13}) und die entsprechenden Bearbeitungszeiten in das ProSiT Ablaufdiagramm übertragen (mNR_7). Da einige Verben nicht im Repository definiert sind, beziehungsweise die Aktivitätsart geändert

wurde, womit kein entsprechender Eintrag vorhanden ist, wird bei den entsprechenden Aktivitäten die Erfassungsmethode messen unterstellt, womit diese Aktivitäten eine lange Bearbeitungszeit erhalten. Da die Bearbeitungszeit nur beispielhaft verwendet wird, ist ein Anspruch einer realitätsgetreuen Abbildung der Bearbeitungszeiten nicht gegebenen, beziehungsweise auch nicht vorgesehen. Das Ergebnis der Normalisierung ist in den Abbildungen 96 und 97 dargestellt.

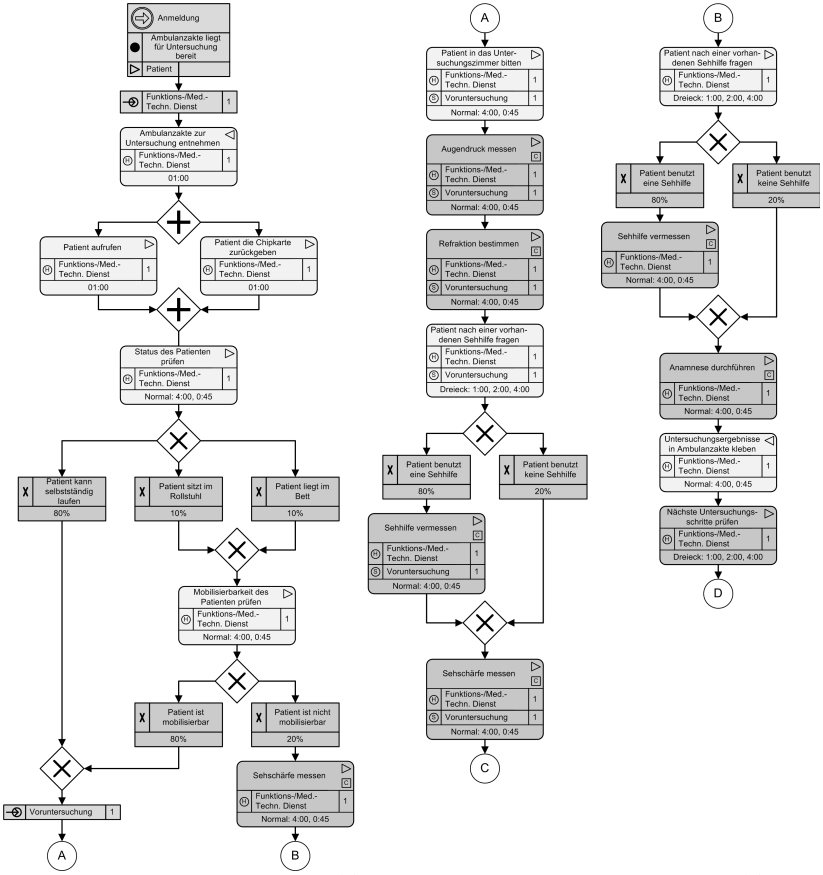


Abbildung 96: Konsistentes ProSiT Ablaufdiagramm der Voruntersuchung – Teil 1

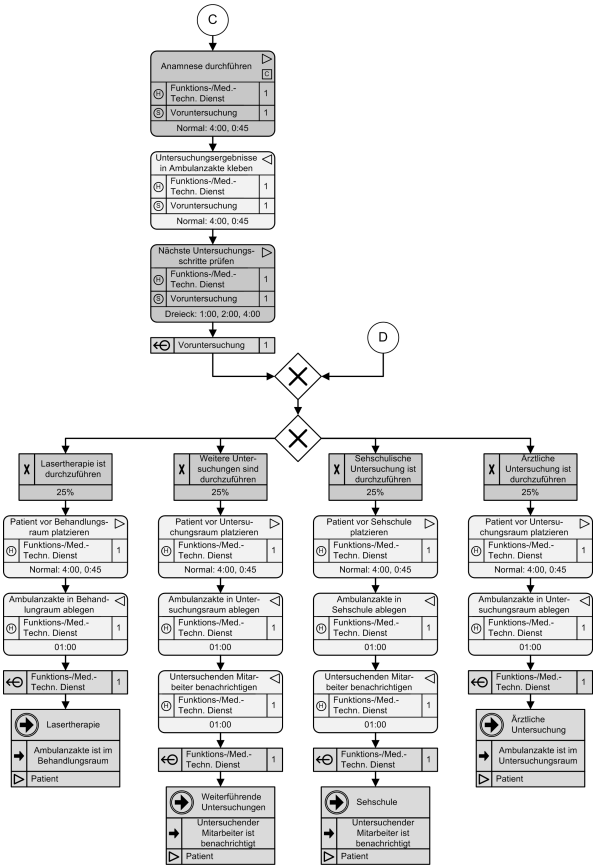


Abbildung 97: Konsistentes ProSiT Ablaufdiagramm der Voruntersuchung – Teil 2

4.4 Evaluation der Überführung des Transformationsmodells in die Zielumgebungen

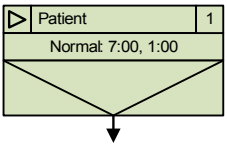
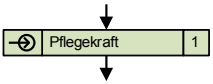
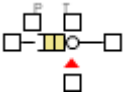
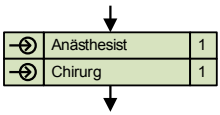
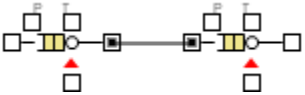
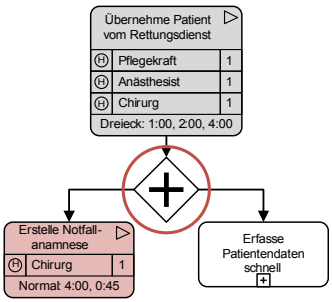
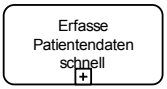
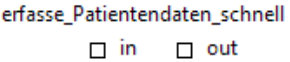
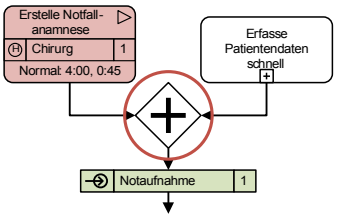
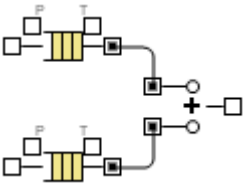
Die beiden konsistenten ProSiT Ablaufdiagramme, der chirurgische Notfall in den Abbildungen 89 und 90 sowie die Voruntersuchung in Abbildung 96 und 97 werden für die Evaluation der Transformationsregeln in die Simulationsumgebungen verwendet. Zuerst werden beide Modelle in die Simulationssoftware AnyLogic überführt und anschließend in die Simulationssoftware Arena.

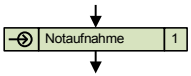
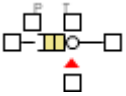
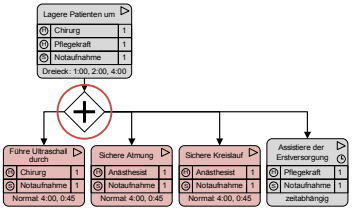
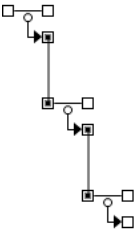
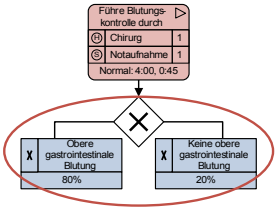
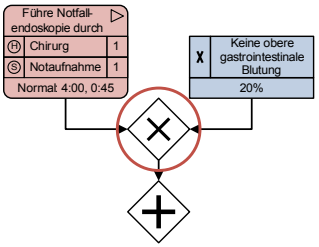
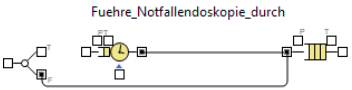
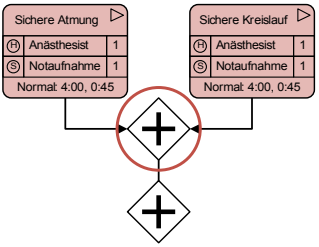
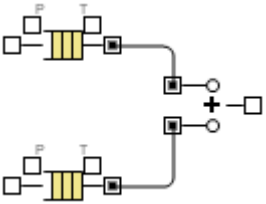
4.4.1 AnyLogic als Zielumgebung

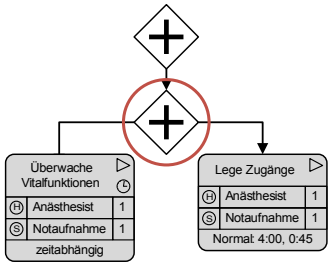
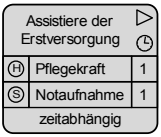
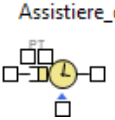
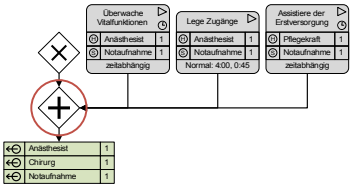
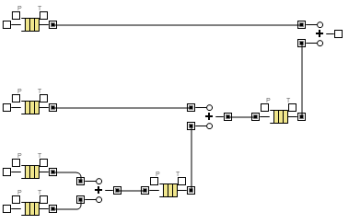
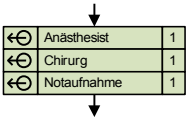
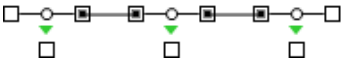
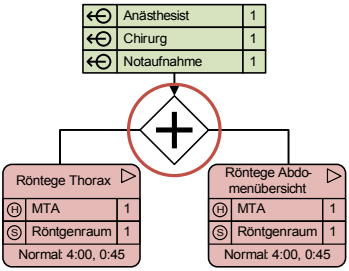
Die Überführung der Voruntersuchung und des chirurgischen Notfalls kann problemlos nach AnyLogic überführt werden. Die Anwendbarkeitsregeln greifen bei beiden Modellen nicht. Nachfolgend werden daher die Transformationen für beide ProSiT Ablaufdiagramme dargestellt.

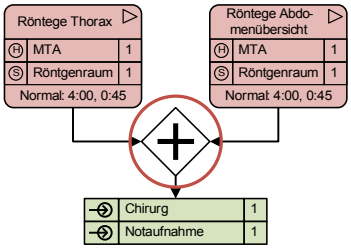
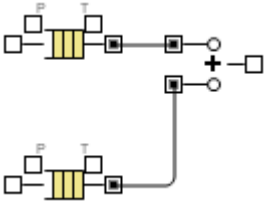
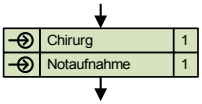
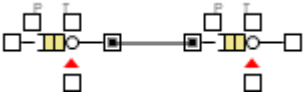
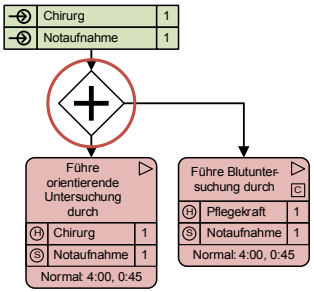
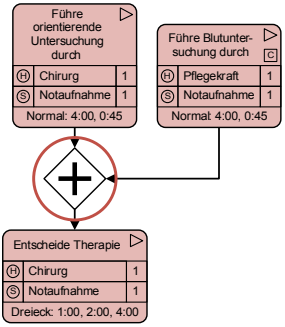
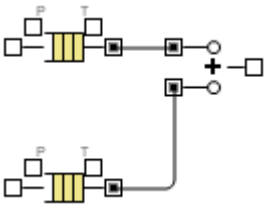
Chirurgischer Notfall
Wie bei der Überführung der Geschäftsprozessmodelle in das ProSiT Ablaufdiagramm wird bei der Überführung in die Simulationsumgebungen ebenfalls die Überführung der Aktivität nicht einzeln aufgeführt. Mit Ausnahme von zeitabhängigen Aktivitäten werden alle Aktivitäten mit der Transformationsregel $alTR_3$ nach AnyLogic überführt. Die Überführung einer Aktivität nach der Transformationsregel $alTR_4$ für zeitabhängige Aktivitäten ist exemplarisch einmal in der nachfolgenden Tabelle 59 mit der Anwendung der restlichen Transformationsregeln dargestellt.

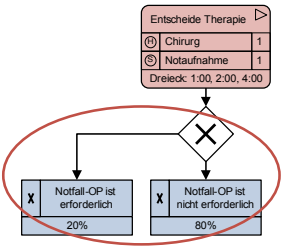
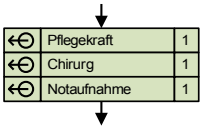
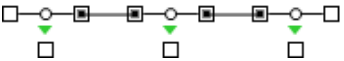
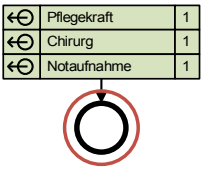
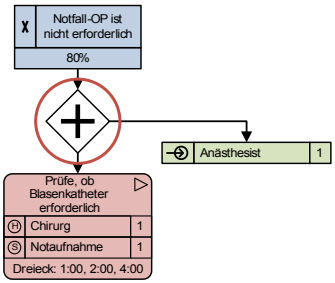
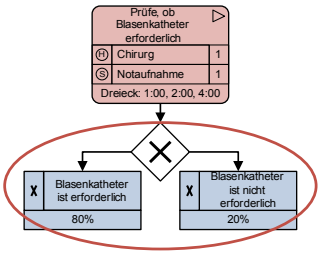
Tabelle 59: Anwendung der AnyLogic Transformationsregeln auf den chirurgischen Notfall

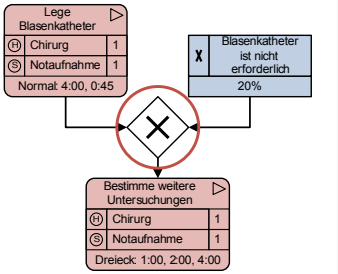
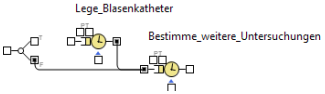
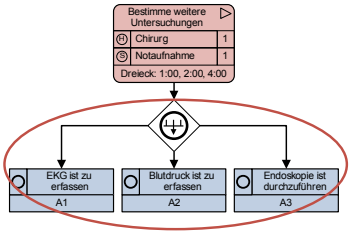
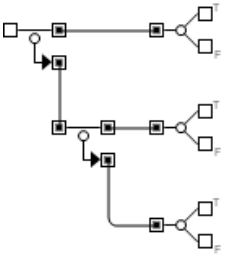
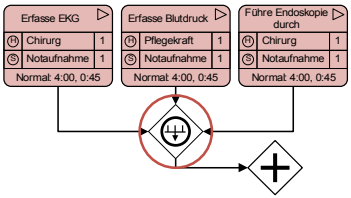
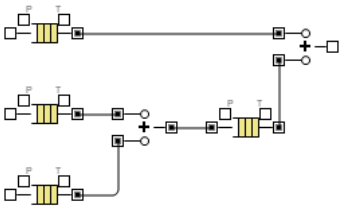
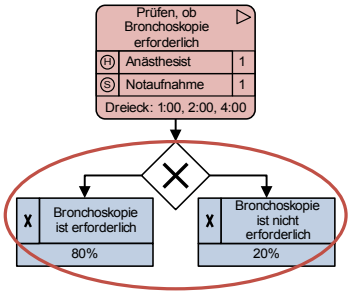
ProSiT Element	alTR	Resultierendes AnyLogic Element
	$alTR_1$	
	$alTR_6$	
	$alTR_6$	
	$alTR_{14}$	
	$alTR_5$	
	$alTR_{15}$	

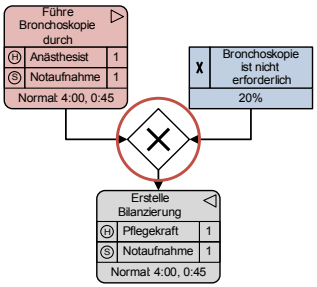
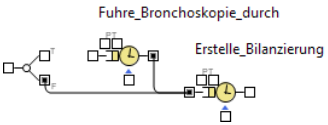
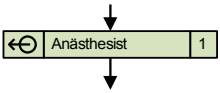
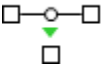
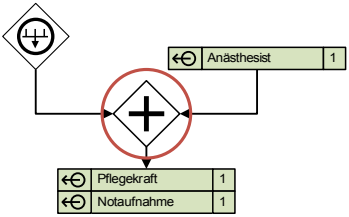
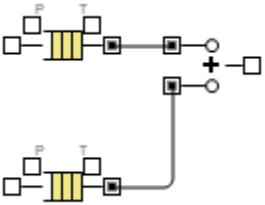
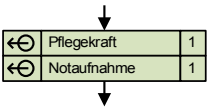
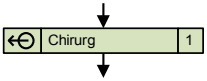
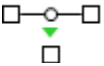
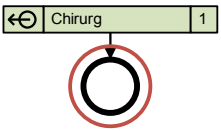
ProSiT Element	alTR	Resultierendes AnyLogic Element
	$alTR_6$	
	$alTR_{14}$	
	$alTR_{18}$	
	$alTR_{19}$	
	$alTR_{15}$	

ProSiT Element	alTR	Resultierendes AnyLogic Element
	$alTR_{14}$	
	$alTR_{14}$	
	$alTR_{15}$	
	$alTR_{7}$	
	$alTR_{14}$	

ProSiT Element	alTR	Resultierendes AnyLogic Element
	$alTR_{15}$	
	$alTR_{6}$	
	$alTR_{14}$	
	$alTR_{15}$	

ProSiT Element	alTR	Resultierendes AnyLogic Element
	$alTR_{18}$	
	$alTR_{7}$	
	$alTR_{2}$	
	$alTR_{14}$	
	$alTR_{18}$	

ProSiT Element	alTR	Resultierendes AnyLogic Element
	$alTR_{19}$	
	$alTR_{25}$	
	$alTR_{22}$	
	$alTR_{18}$	

ProSiT Element	alTR	Resultierendes AnyLogic Element
	$alTR_{19}$	
	$alTR_{7}$	
	$alTR_{15}$	
	$alTR_{7}$	
	$alTR_{7}$	
	$alTR_{2}$	

Das resultierende AnyLogic Simulationsmodell ist in der nachfolgenden Abbildung 98 dargestellt.

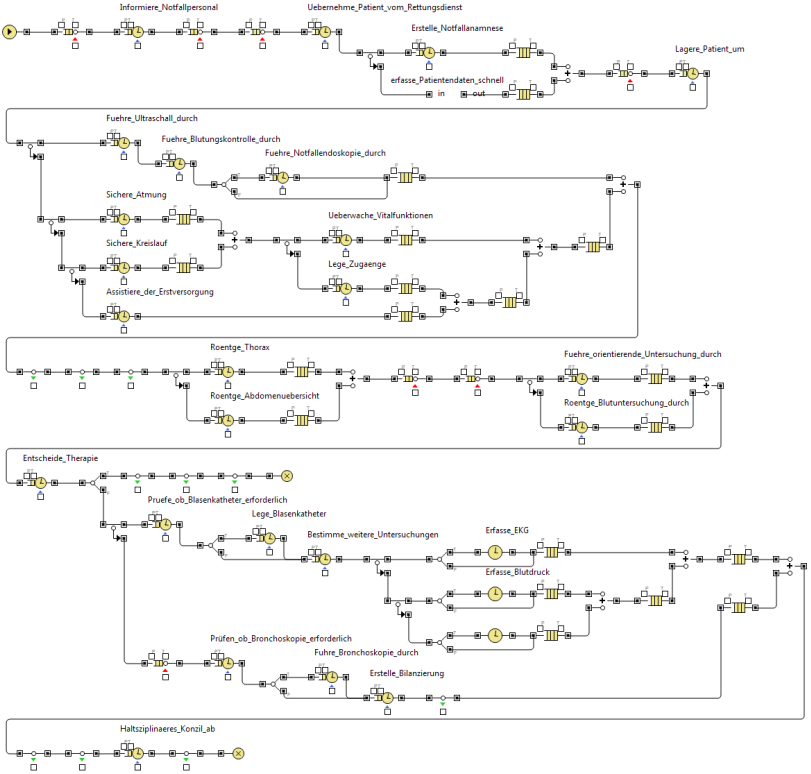
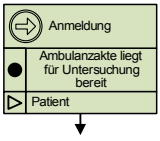
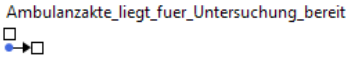
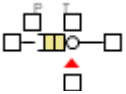
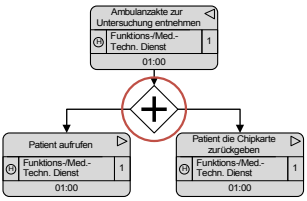
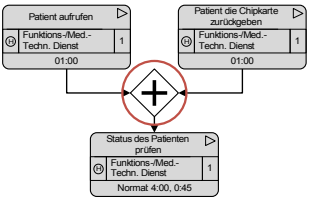
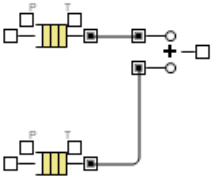
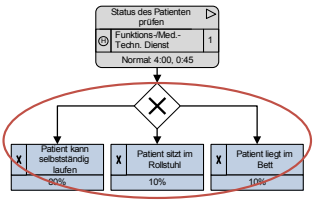


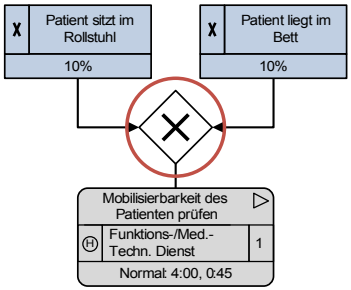
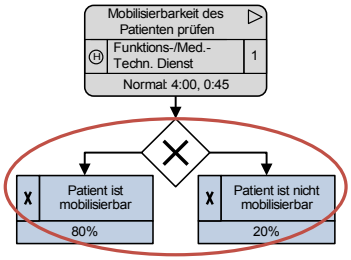
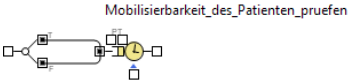
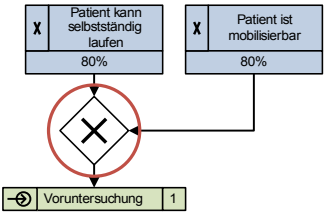
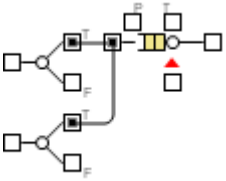
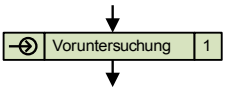
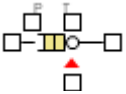
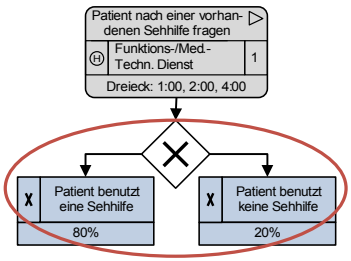
Abbildung 98: Chirurgischer Notfall als Simulationsmodell in AnyLogic

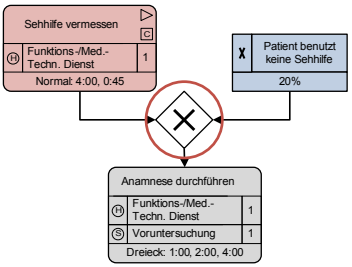
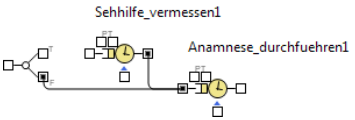
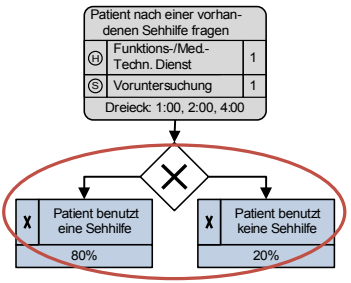
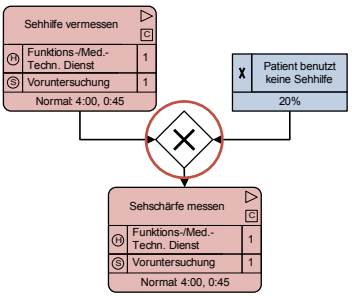
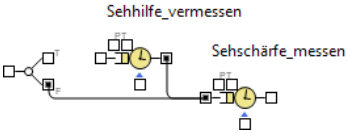
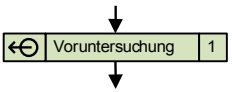
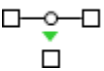
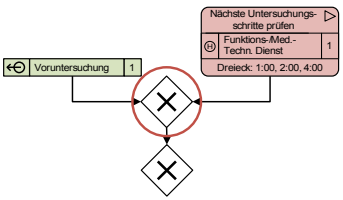
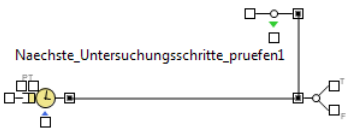
Voruntersuchung

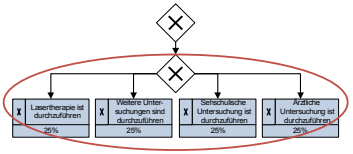
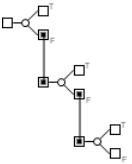
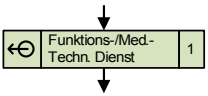
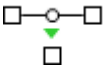
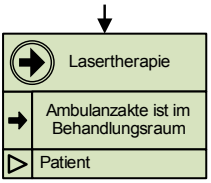
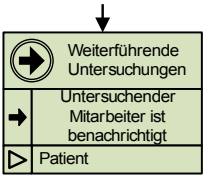
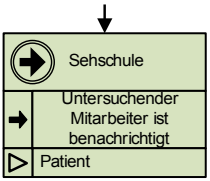
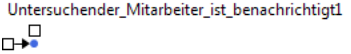
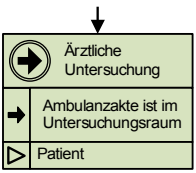
Bei der Voruntersuchung werden alle Aktivitäten mit der Transformationsregel $alTR_3$ nach AnyLogic überführt. Die Anwendung der Transformationsregeln auf die restlichen Elemente ist in Tabelle 60 dargestellt.

Tabelle 60: Anwendung der AnyLogic Transformationsregeln auf die Voruntersuchung

ProSiT Element	alTR	Resultierendes AnyLogic Element
	$alTR_{11}$	
	$alTR_6$	
	$alTR_{14}$	
	$alTR_{15}$	
	$alTR_{18}$	

ProSiT Element	alTR	Resultierendes AnyLogic Element
	$alTR_{19}$	
	$alTR_{18}$	
	$alTR_{19}$	
	$alTR_{6}$	
	$alTR_{18}$	

ProSiT Element	alTR	Resultierendes AnyLogic Element
	$alTR_{19}$	
	$alTR_{18}$	
	$alTR_{19}$	
	$alTR_{7}$	
	$alTR_{19}$	

ProSiT Element	alTR	Resultierendes AnyLogic Element
	$alTR_{18}$	
	$alTR_{7}$	
	$alTR_{10}$	
	$alTR_{10}$	
	$alTR_{10}$	
	$alTR_{10}$	

Das Ergebnis der Transformation ist Abbildung 99 aufgeführt.

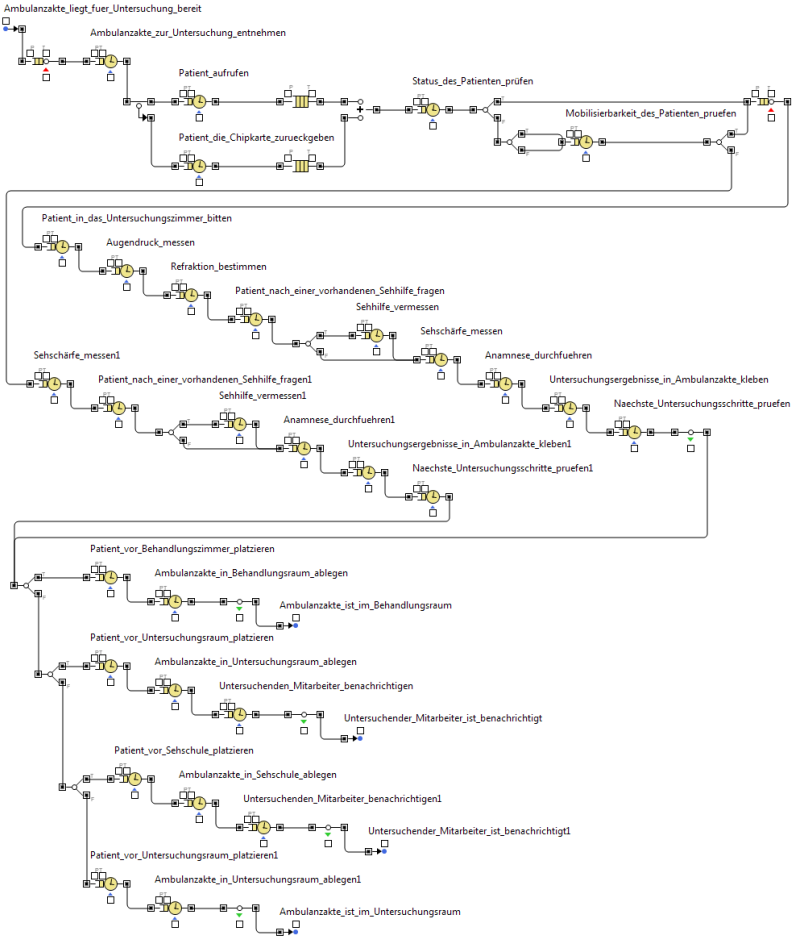


Abbildung 99: Voruntersuchung als Simulationsmodell in AnyLogic

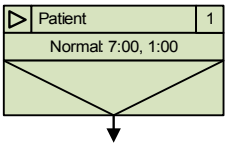
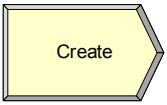
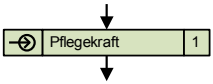
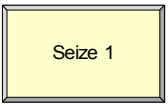
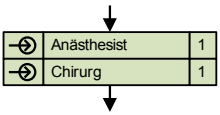
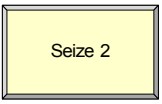
4.4.2 Arena als Zielumgebung

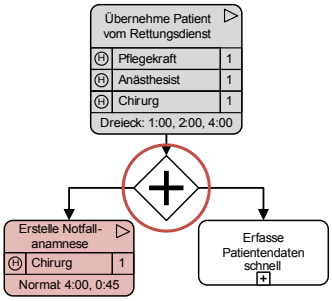
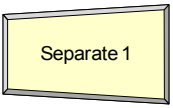
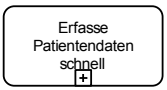
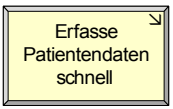
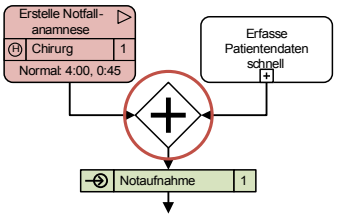
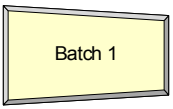
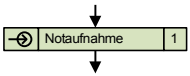
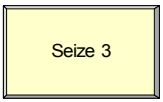
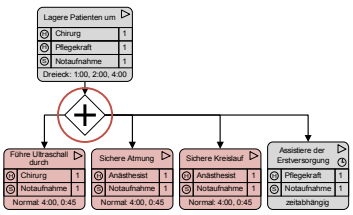
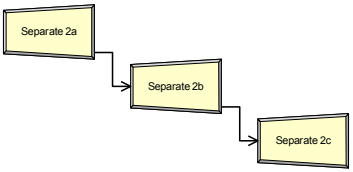
Keine der drei Anwendbarkeitsregeln von Arena (arAR) wird bei den beiden ProSiT Ablaufdiagrammen aktiv. Somit können die Transformationsregeln (arTR) direkt auf den chirurgischen Notfall und die Voruntersuchung angewendet werden.

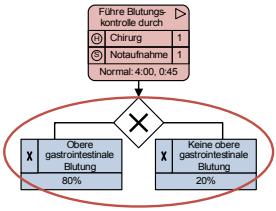
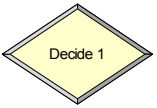
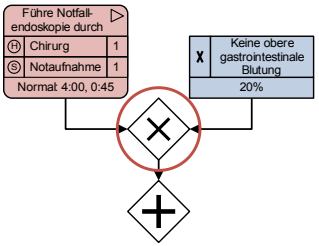
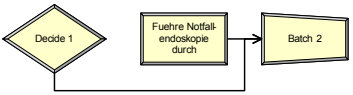
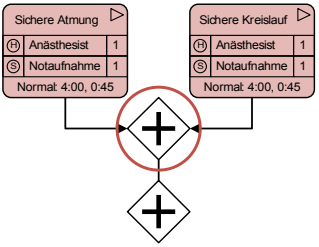
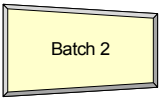
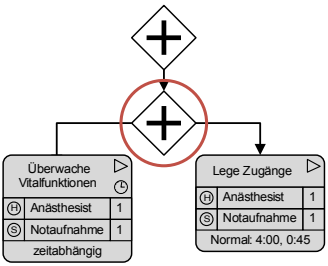
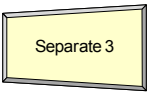
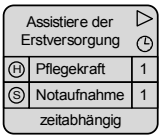
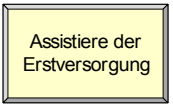
Chirurgischer Notfall

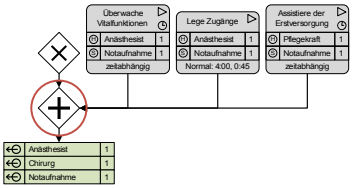
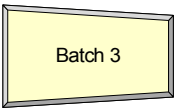
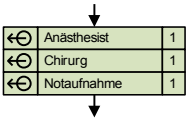
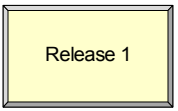
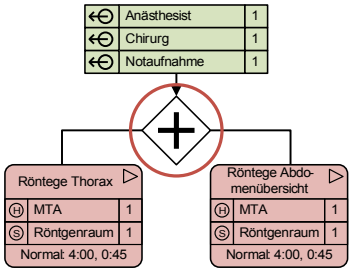
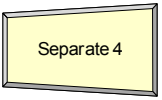
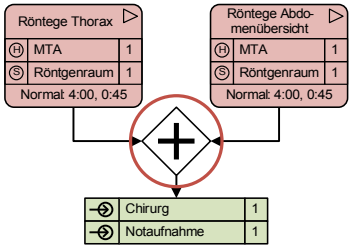
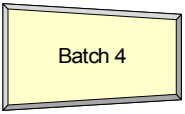
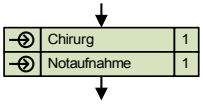
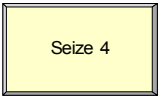
Da die Transformationsregeln nach Arena nach dem gleichen Konzept konzipiert sind, wie die Transformationsregeln nach AnyLogic, werden alle nicht zeitabhängigen Aktivitäten nach der Transformationsregel arTR3 überführt. Die Anwendung der Transformationsregel arTR4 und die der restlichen Transformationsregeln sind in der nachfolgenden Tabelle 61 dargestellt.

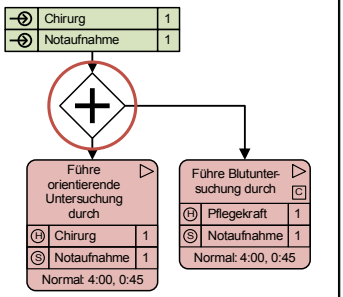
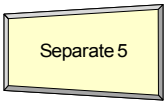
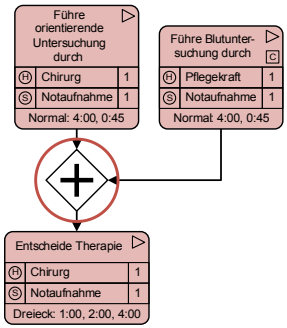
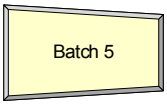
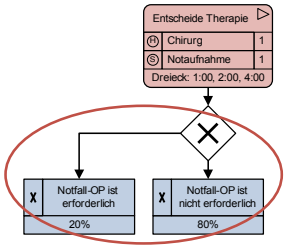
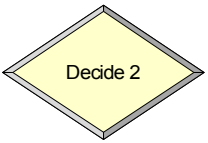
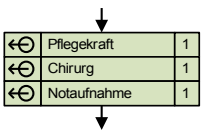
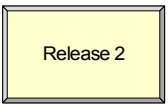
Tabelle 61: Anwendung der Arena Transformation auf den chirurgischen Notfall

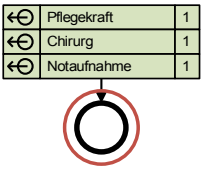
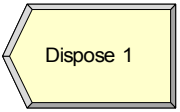
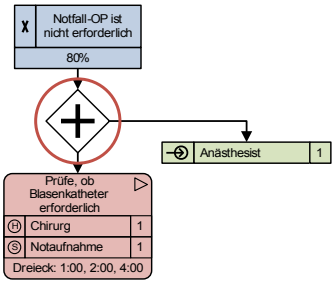
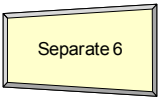
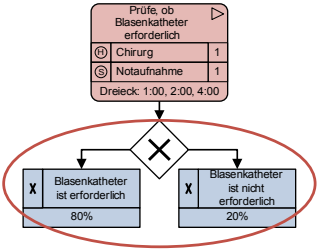
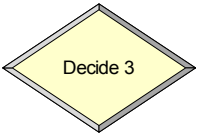
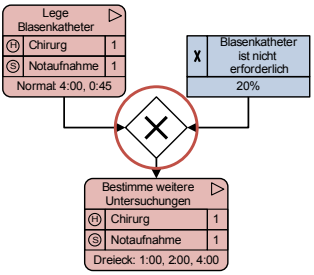
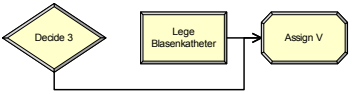
ProSiT Element	arTR	Resultierendes Arena Element
	$arTR_1$	
	$arTR_6$	
	$arTR_6$	

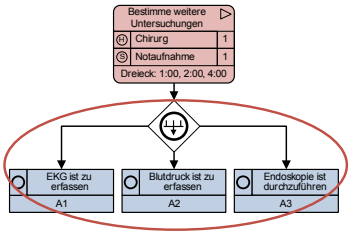
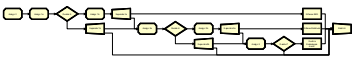
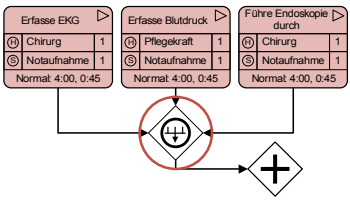
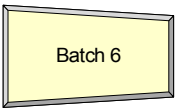
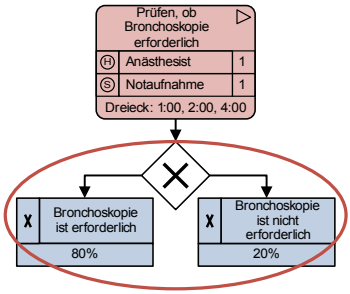
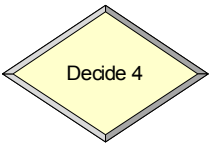
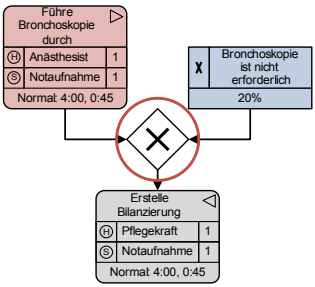
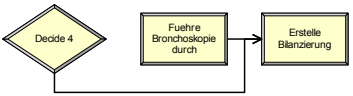
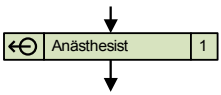
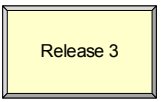
ProSiT Element	arTR	Resultierendes Arena Element
	$arTR_{14}$	
	$arTR_{5}$	
	$arTR_{15}$	
	$arTR_{6}$	
	$arTR_{14}$	

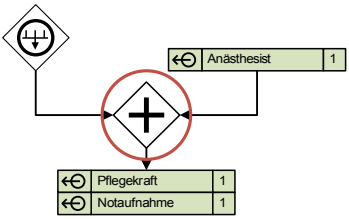
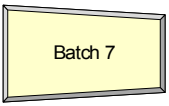
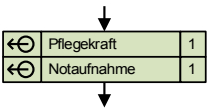
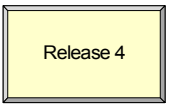
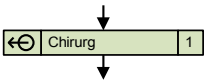
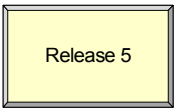
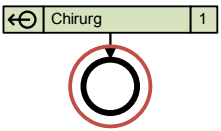
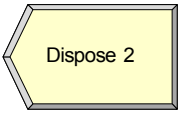
ProSiT Element	arTR	Resultierendes Arena Element
	arTR$_{18}$	
	arTR$_{19}$	
	arTR$_{15}$	
	arTR$_{14}$	
	arTR$_{14}$	

ProSiT Element	arTR	Resultierendes Arena Element
	$arTR_{15}$	
	$arTR_{7}$	
	$arTR_{14}$	
	$arTR_{15}$	
	$arTR_{6}$	

ProSiT Element	arTR	Resultierendes Arena Element
	$arTR_{14}$	
	$arTR_{15}$	
	$arTR_{18}$	
	$arTR_{7}$	

ProSiT Element	arTR	Resultierendes Arena Element
	$arTR_2$	
	$arTR_{14}$	
	$arTR_{18}$	
	$arTR_{19}$	

ProSiT Element	arTR	Resultierendes Arena Element
	$arTR_{25}$	
	$arTR_{22}$	
	$arTR_{18}$	
	$arTR_{19}$	
	$arTR_{7}$	

ProSiT Element	arTR	Resultierendes Arena Element
	$arTR_{15}$	
	$arTR_{7}$	
	$arTR_{7}$	
	$arTR_{2}$	

Das resultierende Simulationsmodell in Arena ist in der nachfolgenden Abbildung 100 dargestellt. Die Transformationsregel $arTR_{25}$, welche das verzweigende inklusive wait-for-all Gateway überführt beinhaltet die gleichen Elemente, wie bei der Erläuterung der Regel aufgeführt wurden. Übersichtlich lässt sich diese aber nicht in Tabelle 61 darstellen. Die betreffenden Elemente sind aber auch in Abbildung 108 zwischen dem „Assign V“ und dem „Batch 6“ Element dargestellt.

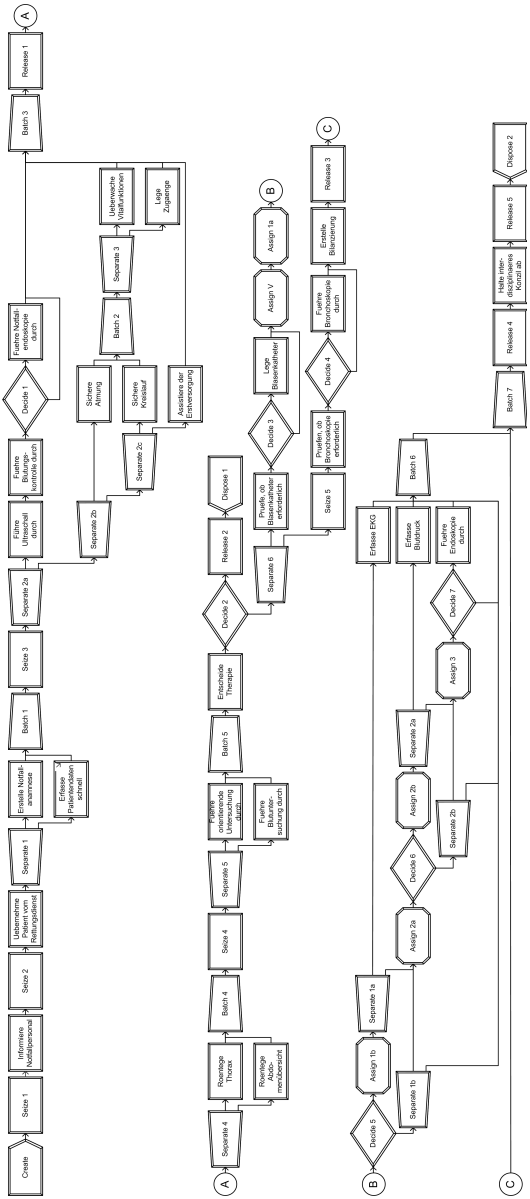
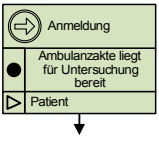
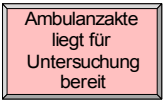
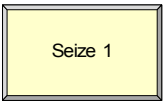
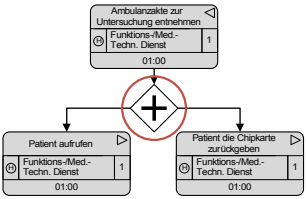
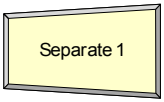
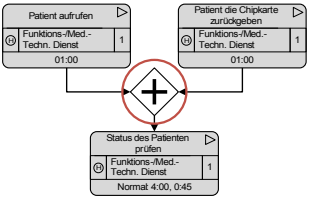
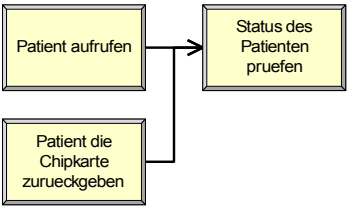
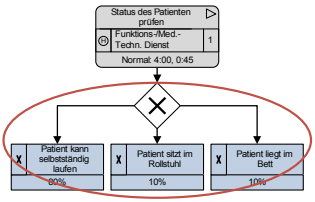
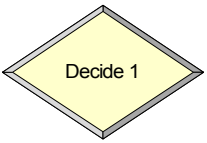


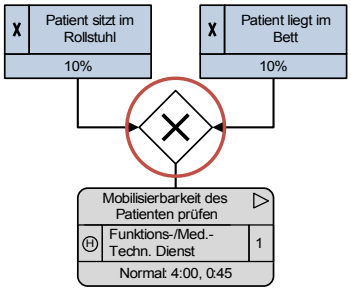
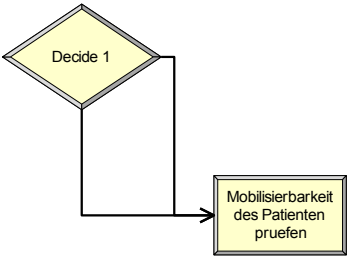
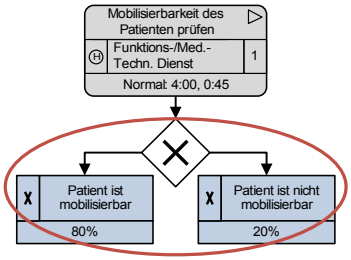
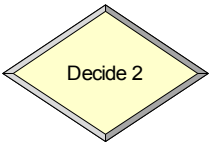
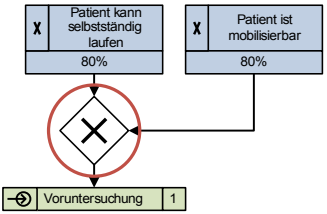
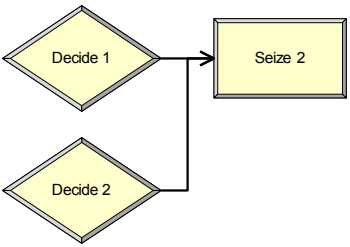
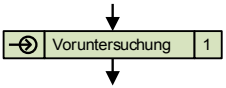
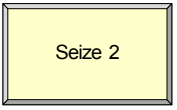
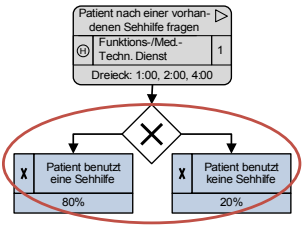
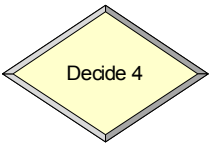
Abbildung 100: Chirurgischer Notfall als Simulationsmodell in Arena

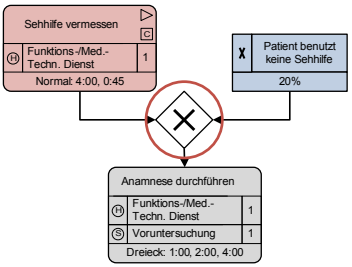
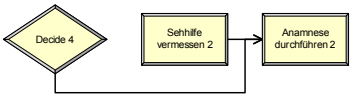
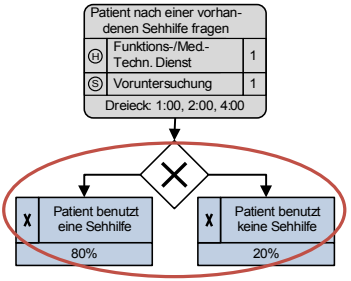
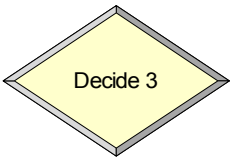
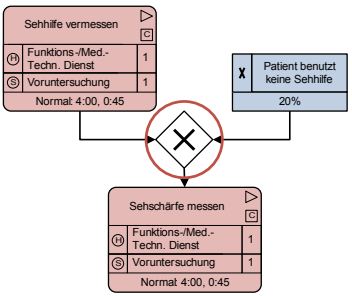
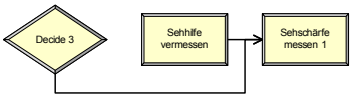
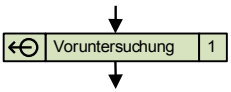
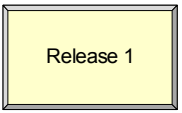
Voruntersuchung

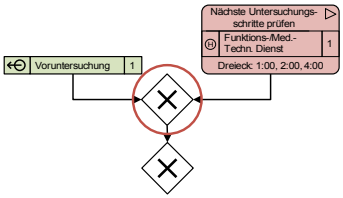
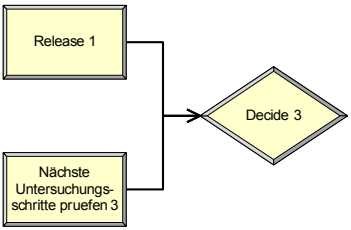
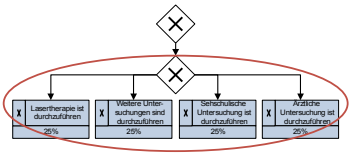
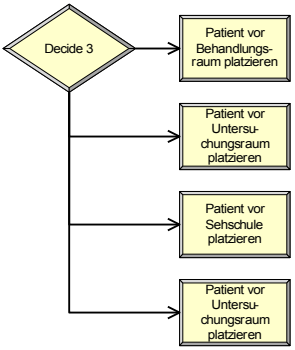
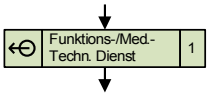
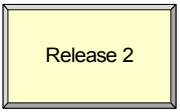
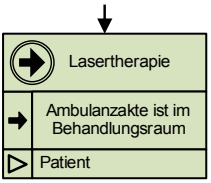
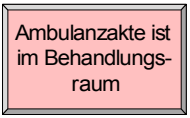
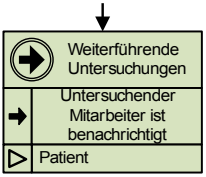
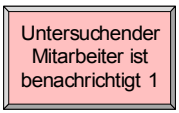
Wie bei der Überführung nach AnyLogic erfolgt die Transformation aller Aktivitäten nach Arena analog mit der Transformationsregel arTR3. Die Anwendung der Transformationsregeln auf die restlichen Elemente ist in der nachfolgenden Tabelle 62 aufgeführt.

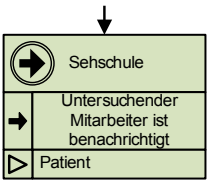
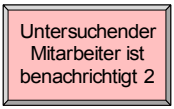
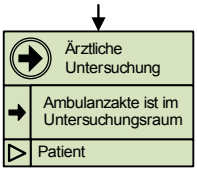
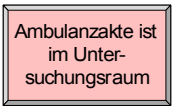
Tabelle 62: Anwendung der Arena Transformationsregeln auf die Voruntersuchung

ProSiT Element	arTR	Resultierendes Arena Element
	$arTR_{11}$	
	$arTR_{6}$	
	$arTR_{14}$	
	$arTR_{15}$	
	$arTR_{18}$	

ProSiT Element	arTR	Resultierendes Arena Element
	$arTR_{19}$	
	$arTR_{18}$	
	$arTR_{19}$	
	$arTR_{6}$	
	$arTR_{18}$	

ProSiT Element	arTR	Resultierendes Arena Element
	$arTR_{19}$	
	$arTR_{18}$	
	$arTR_{19}$	
	$arTR_{7}$	

ProSiT Element	arTR	Resultierendes Arena Element
	$arTR_{19}$	
	$arTR_{18}$	
	$arTR_{7}$	
	$arTR_{10}$	
	$arTR_{10}$	

ProSiT Element	arTR	Resultierendes Arena Element
	$arTR_{10}$	
	$arTR_{10}$	

In Abbildung 101 ist das mit den aufgeführten Transformationsregeln erzeugte Transformationsmodell dargestellt.

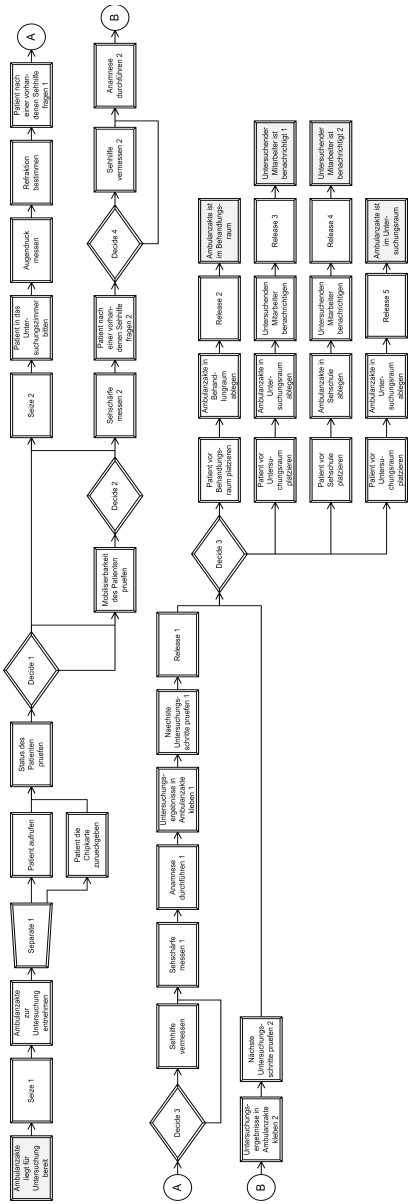
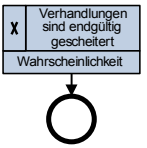
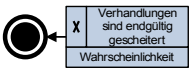
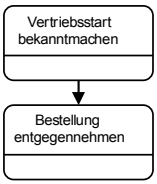
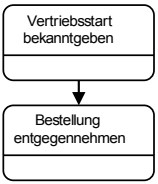
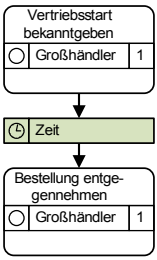


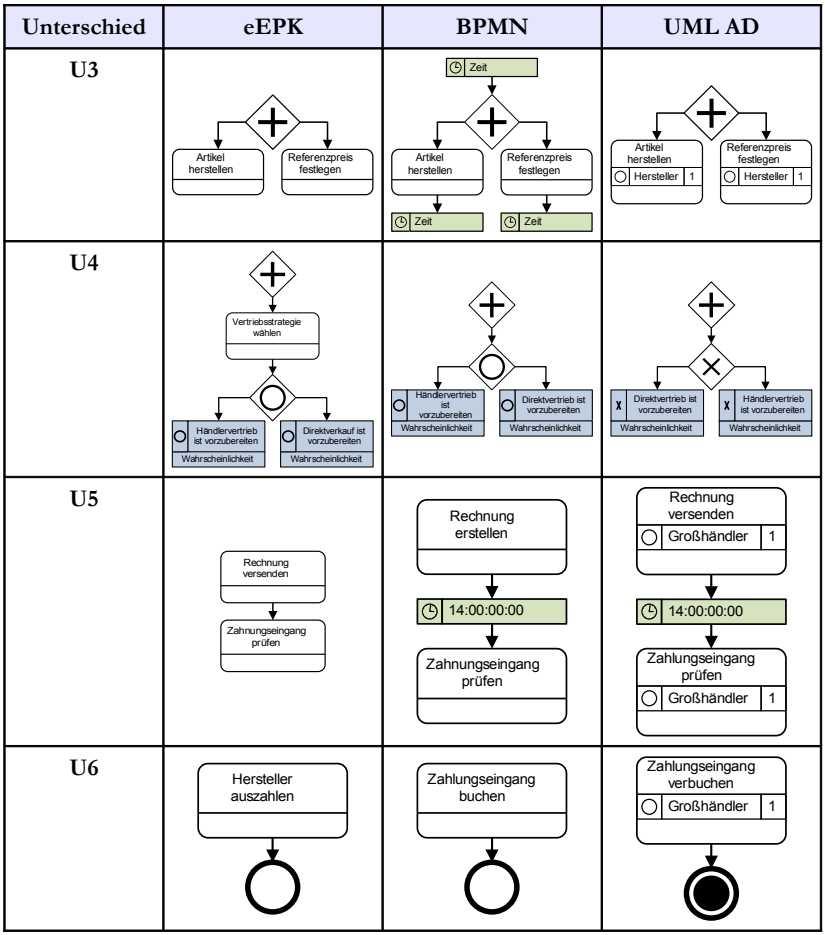
Abbildung 101: Voruntersuchung als Simulationsmodell in Arena

4.5 Diskussion der Evaluation

Die Diskussion der Transformationsregeln von den Quellmodellen zum ProSiT Ablaufdiagramm, soll anhand der Produkteinführung erfolgen, die in allen drei Quellnotationen vorliegt. Eine These, die den Transformationsregeln unterstellt werden kann, besagt, dass wenn die Transformationsregeln korrekt sind, dann müssen diese das gleiche Transformationsmodell erzeugen. Erste Vergleiche zeigten jedoch Unterschiede in den erzeugten ProSiT Ablaufdiagrammen. Zwei mögliche Ursachen können der Grund dafür sein: Fehler liegen in den Transformationsregeln vor oder die Quellmodelle repräsentieren jeweils nicht das gleiche Original. Eine Übersicht aller Unterschiede ist in Tabelle 63 aufgeführt. Beim BPMN Modell ist zu beachten, dass das Modell zum einen aus der eEPK, zum anderen aus dem UML Aktivitätsdiagramm abgeleitet wurde und selbst erstellt ist. Dieser Aspekt ist im nachfolgenden Vergleich stets zu berücksichtigen.

Tabelle 63: Gegenüberstellung der Unterschiede der konzeptionellen Produkteinführung

Unterschied	eEPK	BPMN	UML AD
U1			
U2			



Bei U1 fällt auf, dass sich das erzeugte ProSiT Ablaufdiagramm der eEPK von den anderen beiden Modellen unterscheidet. Anstelle einer terminierenden Senke wird eine normale Senke verwendet. Der Grund hierfür liegt bei der eEPK. Diese berücksichtigt kein terminierendes Endereignis, mit dem alle laufenden Prozessinstanzen abgebrochen werden können. Dem entsprechend erzeugen die Transformationsregeln keine terminierende Senken.

Ein Unterschied, der auf die Transformationsregeln zurückzuführen ist, kann bei U2 gefunden werden. Das BPMN Modell wurde von der eEPK und dem UML Aktivitätsdiagramm abgeleitet. Daher liegt nur eine sequenzielle Abfolge von zwei Aktivitäten vor. Beim UML Aktivitätsdiagramm ist hingegen eine Verzögerung enthalten. Wird das Quellmodell des UML Aktivitätsdiagramms in Abbildung 76 betrachtet, dann ist aber auch nur eine sequenzielle Abfolge vorzufinden. Die Verzögerung ist auf die Vorbereitungsregel $adPR_{1\_6}$ zurückzuführen, welche ein Zeitereignis nach einer Signalsendeaktion einfügt, wenn dessen Nachfolger eine Signalereignisaktion ist. Die Verzögerung berücksichtigt, dass zwischen der Bekanntgabe der Vertriebsstrategie Zeit vergeht, bis eine Bestellung eingeht. Dieser Sachverhalt wird aber beim eEPK Modell, in Abbildung 46, nicht berücksichtigt. Da das BPMN Modell von beiden Modellen abgeleitet wurde, ist dies auch nicht berücksichtigt. Bei beiden Modellen liegt daher ein logischer Fehler vor, da direkt auf das Bekanntgeben der Vertriebsstrategie eine Bestellung eintrifft. Der Unterschied im ProSiT Ablaufdiagramm wird zwar von der Regelbasis erzeugt, der Ursache liegt aber bei den Quellmodellen. Bei der eEPK hätte eine externe Funktion berücksichtigt werden müssen (siehe automatische Normalisierungsregel aNR_1) und beim BPMN Modell fehlt ein Timer Zwischenereignis.

Der zweite Unterschied bei U2 ist ebenfalls auf die Quellmodelle, aber auch auf die Transformationsregeln zurückzuführen. Nur beim Transformationsmodell, welches aus dem UML Aktivitätsdiagramm erzeugt wurde, sind Ressourcen in den Aktivitäten enthalten. Die fehlenden Ressourcen bei der eEPK können auf die Modellierung mit Organisationseinheiten zurückgeführt werden. Da durch die Transformationsregel eTR_{12} nur Stellen in das ProSiT Ablaufdiagramm überführt werden, sind bei diesem keine Ressourcen in den Aktivitäten enthalten. Die Transformationsregel bTR_{13}, zur Überführung von Aufgaben aus BPMN Modellen, berücksichtigt nur Lanes. Da aber im Quellmodell nur Pools verwendet werden (Abbildung 65), die somit auch als eine Art Organisationseinheit aufgefasst werden können, sind beim resultierenden Transformationsmodell ebenfalls keine Ressourcen

enthalten. Bei der Regel $adTR_7$ der UML Aktivitätsdiagramm Transformationsregeln wird jedoch die am tiefsten liegende Partition als Ressource verwendet, wodurch ein Unterschied im resultierenden Transformationsmodell entsteht. Damit durch die Transformationsregel $adTR_7$ das gleiche Resultat hinsichtlich der Ressourcen entsteht, müsste hierfür die höchste Partition ignoriert werden. Das UML Aktivitätsdiagramm unterschiedet aber nicht in ähnliche Elemente wie Organisationseinheiten und Stellen, weshalb die oberste Partition nicht ignoriert wird.

Beim Unterschied U3 unterscheidet sich das aus dem BPMN Modell resultierende Transformationsmodell von den anderen beiden Modellen. Im dargestellten Modellausschnitt sind drei Verzögerungen enthalten, die bei den anderen Modellen nicht vorhanden sind. Erzeugt werden die Verzögerungen durch die drei angewendeten Vorbereitungsregeln, die in Tabelle 43 aufgeführt sind. Diese werden angewendet, da Nachrichtenflüsse und Sequenzflüsse bei der Transformation unterschiedlich behandelt werden. Eine eEPK weist nur eine Art von Fluss vor. Das UML Aktivitätsdiagramm hat zwar auch zwei Arten von Flüssen, den Kontrollfluss und den Objektfluss, diese unterschieden sich aber nicht hinsichtlich der Verwendungsmöglichkeiten, wie dies bei der BPMN der Fall ist. Der Unterschied in den erzeugten ProSiT Ablaufdiagrammen ist daher nicht direkt auf die Transformationsregeln zurückzuführen, sondern auf unterschiedliche Modellierungsmöglichkeiten in den Quellmodellen.

In U4 sind zwei Unterschiede erkennbar. Zum einen gibt es eine Aktivität bei der eEPK, die bei den anderen Modellen nicht enthalten ist, zum anderen weist das UML Aktivitätsdiagramm ein exklusives anstelle eines inklusiven Gateways auf, wie dies bei den anderen beiden Modellen der Fall ist. Der Grund für die zusätzliche Funktion bei der eEPK liegt in der fehlenden Entscheidungskompetenz von Ereignissen, weshalb ein weitere Funktion eingefügt werden muss (Overhage 2011, S. 751), die beim UML Aktivitätsdiagramm und bei der BPMN nicht notwendig ist. Der Grund für den Unterschied liegt bei den Spezi-

fikationen der Quellmodelle. Beim zweiten Unterschied ist zunächst zu klären, ob ein inklusives oder exklusives Gateway verwendet werden sollte. Aus semantischer Sicht ist ein inklusives Gateway notwendig, da sowohl ein Direktvertrieb als auch ein Händlervertrieb sich nicht gegenseitig ausschließen. Das exklusive Gateway, welches durch das UML Aktivitätsdiagramm aufgrund der Transformationsregel adTR$_1$2 erzeugt wird, ist somit ein semantischer Fehler. Wie jedoch in den Spezifikationen zum UML Aktivitätsdiagramm aufgeführt wird (OMG 2009, S. 360), handelt es sich bei einer Verzweigung um eine exklusive Verzweigung, weshalb diese in ein exklusives Gateway transformiert wird. Der semantische Fehler ist somit nicht auf die Transformationsregeln zurückzuführen, sondern auf das modellierte UML Aktivitätsdiagramm.

Beim letzten Unterschied U6 sind wie bei U1 unterschiedliche Senken dargestellt. Im Gegensatz zu U1 wurde beim BPMN Modell aber kein terminierendes Endereignis verwendet. Aufgrund dieses Unterschieds im Quellmodell scheint nur das UML Aktivitätsdiagramm unterschiedlich zu sein. Bei einem terminierenden Endereignis im BPMN Quellmodell, wäre jedoch wieder das ProSiT Ablaufdiagramm der eEPK, wie im Unterschied U1 von den anderen beiden Modellen, verschiedenen. Der Unterschied ist daher auf die Modellierung eines unterschiedlichen Sachverhalts im Quellmodel zurückzuführen.

Wie diese Ausführungen darlegen, sind alle Unterschiede in den resultierenden ProSiT Ablaufdiagrammen auf die Quellmodelle zurückzuführen und nicht auf die Transformationsregeln. Lediglich die unterschiedliche Umsetzung der Ressourcen in den Ablaufdiagrammen könnte auf die Regeln zurückgeführt werden. Da das UML Aktivitätsdiagramm aber nur Partitionen unterstützt und bei diesen stets die tiefste für die Ressource verwendet wird, liegt hier lediglich ein Unterschied bei den Ressourcen, die eine Eigenschaft der Aktivitäten sind, vor. Werden die Attribute nicht berücksichtigt, so kann aus der Elementsicht kein Unterschied festgestellt werden, weshalb die These aufgestellt wird, dass die dargelegten Transformationsregeln das gleiche

ProSiT Ablaufdiagramm erzeugen, wenn mit den Quellmodellen das gleiche Original modelliert wurde.

Diskussion der erzeugten Simulationsmodelle
Die dargelegte Transformation des chirurgischen Notfalls sowie der Voruntersuchung in die beiden Simulationsumgebungen umfasst lediglich die Überführung des ProSiT Ablaufdiagramms. Während aber der chirurgische Notfall direkt simuliert werden kann, ist dies bei der Voruntersuchung nicht der Fall. Grund hierfür sind die Referenzen zu den anderen Modellen, wodurch mehrere Simulationsmodelle notwendig sind. Dieser Aspekt beeinflusst die Evaluation aber nicht, da der Schwerpunkt auf der Modellgenerierung liegt.

Werden die Transformationsregeln zu beiden Simulationsumgebungen gegenübergestellt, dann wird festgestellt, dass diese äquivalent formuliert sind. Mit dem ProSiT Ablaufdiagramm als Ausgangspunkt überführt die erste Transformationsregel jeweils die Quelle in die Simulationsumgebung. Durch diese Konzeption könnte beispielsweise das AnyLogic Simulationsmodell in ein Arena Simulationsmodell überführt werden.

Bei einem Probelauf der Voruntersuchung konnte jedoch ein Problem bei AnyLogic identifiziert werden. Während in Arena eine Synchronisation von Marken mit der gleichen Seriennummer erfolgt, liegt dieses Verhalten bei AnyLogic nicht vor. Das Combine Element in AnyLogic verknüpft zwei beliebige Marken und nicht Marken der gleichen Instanz. Inwieweit dieser Aspekt Auswirkungen auf die Simulationsergebnisse hat, muss weiterführend untersucht werden. Ähnliche Probleme liegen aber auch bei den Simulationsumgebungen Plant Simulation und Simul8 vor, bei denen die Zusammenführung von mehreren Marken nicht über eine Seriennummer gesteuert wird. Da jedoch AnyLogic eine weitgehend freie Programmierung mit Java erlaubt, kann dieser Sachverhalt, wie auch bei den anderen Simulationsumgebungen, behoben werden, wenn entsprechende Methoden für das gewünschte Verhalten programmiert werden. Anhängig von der

Bewertung dieses Aspekts kann geschlussfolgert werden, dass entweder die dargelegte Überführung des ProSiT Transformationsmodells zu AnyLogic ein gültiges simulationsfähiges Modell erzeugt oder dass das Modell nicht gültig ist. Unabhängig von diesem Aspekt liegt aber ein simulationsfähiges Modell vor, die Transformationsregeln müssen jedoch noch auf Attributsebene weiter spezifiziert werden.

Bei der Überführung nach Arena und anschließenden Probeläufen konnten keine Probleme ausfindig gemacht werden. Bei diesen Transformationsregeln wird geschlussfolgert: Sie erzeugen ein gültiges simulationsfähiges Modell.

Gesamtbetrachtung der Evaluation

Wird die Evaluation als Ganzes betrachtet, so lässt sich wie eingangs formuliert, nicht von einer strengen Falsifikation sprechen. Hierfür wäre eine Vielzahl an Modellen notwendig, die jeweils mit anderen Strukturen modelliert werden. Gemäß Popper ist es nicht dienlich ähnliche Modelle für die Evaluation heranzuziehen, da auf diese immer die gleichen Regeln angewendet werden. Durch die detaillierte Darstellung der einzelnen Transformationsregeln, sprengt dies aber den Rahmen jeder Arbeit, sodass nur das Endergebnis im Rahmen einer wissenschaftlichen Arbeit dargestellt werden könnte, mit einem entsprechenden Verweis auf die Quelldaten, die zum Ergebnis führten. Durch die Betrachtung von nur jeweils zwei Modellen im Rahmen der Evaluation der einzelnen Regeln, kann nur von einer einfachen Falsifikation gesprochen werden, bis hin zu einer einfachen Anwendung der Regelbasis. Da die Anwendung der Regeln jeweils zu einem gültigen Modell führten, wird dennoch diese Anwendung als Teil der Evaluation der Regelbasis angesehen.

Für die Evaluation sind aber auch gleiche Geschäftsprozesse interessant, die mit unterschiedlichen Modellierungsnotationen erstellt wurden. Dies erlaubt einen direkten Vergleich der erzeugten konzeptionellen Ablaufdiagramme. Aus diesem Vergleich, wie dieser mit dem Geschäftsprozess der Produkteinführung vollzogen wurde,

eignet sich zum einen für die Überprüfung der bestehenden Regeln, aber auch für die Weiterentwicklung der Regelbasis. Dies ist aber nur der Fall, wenn die modellieren Prozessmodelle korrekt sind und Unterschiedene in den erzeugten konzeptionellen Transformationsmodellen auf die Regelbasis zurückzuführen sind. Damit aber ein Geschäftsprozessmodell in unterschiedlichen Notationen vorliegt, müsste dieses selbst modelliert werden. Hierbei besteht dadurch aber die mögliche Gefahr, dass nur Modelle erstellt werden, die für die Bestätigung der Regelbasis erstellt werden. Dementsprechend sind nur solche Modelle für eine Falsifikation und Evaluation geeignet, die erstellt werden ohne direkten Bezug auf das ProSiT Konzept. Da für ein Geschäftsprozess aber nur ein Geschäftsprozessmodell modelliert und dieser in der Regel nicht mit verschiedenen Geschäftsprozessmodellierungsnotationen erstellt wird, ist ein Vergleich mit unterschiedlichen Modellen des gleichen Geschäftsprozesses nur möglich, wenn selbst eine Modellierung vorgenommen wird.

5 Zusammenfassung und offene Forschungsfragen

5.1 Beantwortung der Forschungsfragen und Prüfung der Hypothesen

Die erste Forschungsfrage hat eine zentrale Instanz als Gegenstand, mit dem Geschäftsprozessmodelle in Simulationsmodelle überführt werden können:

Wie können Geschäftsprozessmodelle mittels einer zentralen Instanz in Simulationsmodelle überführt werden?

Der im Kontext dieser Forschungsfrage konzipierte Ansatz ist das ProSiT Konzept, dass sich aus dem Transformationsmodell sowie der Regelbasis aufbaut. Das Transformationsmodell besteht aus dem ProSiT Ablaufdiagramm, der Tätigkeitssicht mit dem Repository sowie der Objektsicht und der Ressourcensicht. Die Überführung erfolgt mittels der Regelbasis. Für Quellmodelle werden drei Regelsätze definiert, welche die Syntaxregeln, die Vorbereitungsregeln und die Transformationsregeln umfassen. Jedes Quellmodell, welches den Syntaxregeln entspricht, kann mit den Vorbereitungs- und Transformationsregeln in das ProSiT Ablaufdiagramm überführt werden. Die Überführung in die Simulationsmodelle, beziehungsweise in die Simulationsumgebungen umfasst Anwendbarkeitsregeln, die ähnlich der Syntaxregeln bestimmen, wann ein Transformationsmodell überführt werden kann. Die eigentlichen Transformationsregeln erzeugen das Simulationsmodell.

Durch die Betrachtung von drei Notationen für die Geschäftsprozessmodellierung und zwei Simulationsumgebungen wurden fünf Regelbasen erstellt. Durch die zentrale Instanz sind weniger Regelbasen notwendig als bei einer direkten Transformation. Bei einer direkten

Transformation wären für die betrachteten Quell- und Zielmodelle sechs Regelsätze notwendig. Jede weitere berücksichtigte Notation oder Simulationsumgebung bedeutet eine größere werdende Aufwandsreduzierung, wenn eine Überführung einer Geschäftsprozessmodellierungsnotation in alle betrachteten Simulationsumgebungen möglich sein soll.

Die zweite Forschungsfrage fokussiert auf die Anpassung und Vorbereitung des aus den Geschäftsprozessmodellen erzeugten Transformationsmodells für die Überführung in die Simulationsmodelle beziehungsweise in die Simulationsumgebungen:

> Wie kann das Transformationsmodell angepasst werden,
> damit es in ein Simulationsmodell überführt werden kann?

Diese Forschungsfrage wird durch die Regelbasis adressiert. Hierfür wurden Normalisierungsregeln konzipiert, die ein konzeptuelles Transformationsmodell in ein konsistentes Transformationsmodell umwandeln. Diese Normalisierung umfasst drei Arten: automatische und semiautomatische Normalisierungsregeln sowie eine manuelle Normalisierung. Durch semiautomatische aber insbesondere automatische Normalisierungsregeln können einige Normalisierungsschritte automatisch ausgeführt werden, wodurch sich der Vorbereitungsaufwand reduziert.

Neben der Betrachtung der Ergebnisse der Arbeit aus Sicht der Forschungsfragen werden nachfolgend die Ergebnisse aus Sicht der fünf Zielstellungen dargelegt:

1. Ein Transformationsmodell entwickeln, um Geschäftsprozessmodelle in Simulationsmodelle zu überführen.

2. Transformationsregeln herleiten, um Geschäftsprozessmodelle automatisch in das Transformationsmodell zu überführen.

3. Regeln und Methoden aufstellen, um das Transformationsmodell mit simulationsrelevanten Daten anzureichern und auf die wesentlichen Aspekte zu reduzieren.

4. Wortarten in den Bezeichnungen des Transformationsmodells untersuchen, um daraus Erkenntnisse für die Beschaffung simulationsrelevanter Daten abzuleiten.

5. Transformationsregeln herleiten, um das Transformationsmodell automatisch in Simulationsumgebungen zu überführen.

Die erste Zielstellung der Arbeit wurde mit der Konzeption des ProSiT Konzepts, welches das ProSiT Ablaufdiagramm und die drei Sichten umfasst, erreicht. Dieser Teil des ProSiT Konzepts ist als Transformationsmodell aufzufassen. Die Herleitung von Transformationsregeln, im Sinne der zweiten Zielsetzung, wurde für die eEPK, den BPMN Prozess und das UML Aktivitätsdiagramm vollzogen. Die Transformationsregeln betrachten hierfür ein Element in seinem semantischen Kontext, die durch Unterstützung der Syntax- und Vorbereitungsregeln eine automatische Transformation realisieren können. Diese automatische Transformation ist im Rahmen der Evaluation dargelegt, bei der für jedes Element angegeben wird, mit welcher Regeln dieses in das Transformationsmodell überführt wird.

Die dritte Zielstellung geht einher mit der zweiten Forschungsfrage und kann durch Heranziehen der Normalisierungsregeln als erreicht deklariert werden. Die Regelbasis der Normalisierung umfasst Regeln, die das Transformationsmodell zum einen erweitern, zum anderen auf wesentliche Aspekte reduzieren.

Die vierte Zielstellung hingegen wurde nur in Ansätzen näher untersucht und stellt hauptsächlich ein Nebenprodukt der Forschungsarbeit dar. Abgedeckt wird dieser Aspekt durch die linguistischen Normalisierungsregeln, die bei den automatischen vorzufinden sind. Eine weitere Berücksichtigung von Wortarten, in Form von Verben

erfolgt in der Tätigkeitssicht mittels des Repositorys. Eine tiefer gehende Untersuchung von Wortarten in den Bezeichnungen der Elemente ist aber als offener Forschungspunkt aufzuführen, wofür aber bereits erste Grundlagen gelegt sind.

Die fünfte Zielstellung ist äquivalent zur zweiten Zielstellung aufzufassen. Für die Überführung wurden Anwendbarkeitsregeln und Transformationsregeln konzipiert, die ein simulationsfähiges Modell aus einem konsistenten Transformationsmodell erzeugen. Durch die Darlegung in der Evaluation kann es als belegt angesehen werden, dass eine Transformation automatisch erfolgen kann. Die Transformationsregeln zu Arena erzeugen ein korrektes Simulationsmodell, während die Transformationsregeln zu AnyLogic noch auf Attributsebene weiter spezifiziert werden müssen. Bezogen auf Elementebene wird aber die These aufgestellt, dass ein korrektes Simulationsmodell erzeugt wird.

Im Rahmen der Zielstellung der Arbeit wurden neben den Forschungsfragen und den fünf Zielstellungen auch fünf Hypothesen formuliert, die nachfolgend betrachtet werden. Die ersten drei Hypothesen müssen hierbei jeweils aus Sicht der einzelnen Quell- oder Zielmodells betrachtet werden.

H1 Wenn alle Elemente des Geschäftsprozessmodells in ein Transformationsmodell überführt werden können, dann ist das Transformationsmodell eine semantische Obermenge der betrachteten Geschäftsprozessmodellnotationen.

H2 Wenn alle Elemente des Transformationsmodells in die Notation der Simulationsmodelle überführt werden können, dann ist das Transformationsmodell eine semantische Obermenge der betrachteten Simulationsmodelle.

H3 Wenn im Kontext des Transformationsmodells jedes semantische Konstrukt einer Quellsprache in ein semantisch identisches Konstrukt einer Zielsprache überführt werden kann, dann kann die Transformation automatisch erfolgen.

Die erste Hypothese fordert, dass alle Elemente des Geschäftsprozessmodells in das Transformationsmodell überführbar sein müssen, damit von einer semantischen Obermenge gesprochen werden kann. Bezogen auf die drei betrachteten Notationen, trifft diese Hypothese nur auf die eEPK zu. Diese definiert keine weiterführenden Syntaxregeln, wodurch alle betrachteten Modellelemente in das ProSiT Ablaufdiagramm überführt werden können. Zu beachten ist, dass es sich auf die betrachteten Modellelemente gemäß des EPML Formats bezieht. Da Erweiterungen der eEPK der Funktion angehängt werden und nicht als Flusselemente dienen, kann die Hypothese aber auch im Hinblick auf weitere Elemente als wahr angesehen werden.

Für den BPMN Prozess und das UML Aktivitätsdiagramm muss die Hypothese als falsifiziert angesehen werden. Der Grund liegt aber bei der Kompatibilität zu den Simulationsumgebungen. Durch eine Erweiterung des ProSiT Ablaufdiagramms können die ausgeschlossenen Elemente der BPMN und des UML Aktivitätsdiagramms ebenfalls überführt werden, wodurch dann die Hypothesen als wahr angesehen werden könnten. Beispielhaft wurde dies mit dem terminierenden Endereignis in BPMN beziehungsweise mit dem Aktivitätsende des UML Aktivitätsdiagramms vollzogen. Diese Elemente werden in eine terminierende Senke überführt. Die terminierende Senke kann aber nicht in die beiden betrachteten Simulationsumgebungen überführt werden. Hierdurch entsteht ein Konflikt mit der zweiten Hypothese, dass wenn ein Element einer Geschäftsprozessmodellierungsnotation nicht von einer Simulationsumgebung unterstützt wird, dieses entweder durch eine Syntaxregel oder eine Anwendbarkeitsregel ausgeschlossen werden muss, wodurch eine der beiden Hypothesen für eine Geschäftsprozessmodellierungsnotation oder Simulationsumgebung als falsifiziert anzugeben ist.

Aufgrund der terminierenden Senke ist die zweite Hypothese für beide Simulationsumgebungen als falsifiziert aufzufassen. Wird dieses Element jedoch ausgeschlossen, dann wäre die Hypothese für die Simulationsumgebung Arena wahr, für AnyLogic jedoch dennoch falsifiziert, da für AnyLogic weitere Anwendbarkeitsregeln definiert sind. Strenge Anwendbarkeitsregeln sowie auch Syntaxregeln sind damit entscheidend, ob die ersten zwei Hypothesen als wahr oder als falsifiziert einzuordnen sind.

Die Bewertung der dritten Hypothese ist abhängig von der Sichtweise auf die Syntax- und Anwendbarkeitsregeln. Wenn die semantischen Konstrukte nach Anwendung dieser Regeln betrachtet werden, dann kann die Transformation automatisch erfolgen, wie dies im Rahmen der Evaluation dargelegt wurde und somit die Hypothese teilweise als wahr angesehen werden. Andernfalls muss, da Elemente ausgeschlossen werden, die Hypothese als falsifiziert aufgefasst werden. Bezogen auf diese beiden Fälle wird der erste Fall als gültig angesehen. Hierfür soll eine Hilfeannahme der Hypothese hinzugefügt werden:

H3a Wenn im Kontext des Transformationsmodells, jedes semantische Konstrukt, dass nicht durch Syntax- oder Anwendbarkeitsregeln ausgeschlossen wird, einer Quellsprache in ein semantisch identisches Konstrukt einer Zielsprache überführt werden kann, dann kann die Transformation automatisch erfolgen.

Durch das Hinzufügen dieser Hilfsannahme können die einzelnen Transformationsregeln betrachtet werden. Für die Hypothese H3 beziehungsweise die neue Hypothese H3a ist das entscheidende Wort „identisch“. Dies trifft auf alle Transformationsregeln zu mit Ausnahme der Transformationsregeln nach AnyLogic. Wie bei der Diskussion der Evaluationsergebnisse aufgeführt wird, sind auf Attributsebene weitere Methoden zu definieren, damit das Verhalten umgesetzt wird, wie dieses vom ProSiT Ablaufdiagramm vorgesehen wird. Bezogen auf AnyLogic ist die Hypothese H3a als falsifiziert

anzusehen, während bei den Transformationsregeln von der eEPK, der BPMN, dem UML Aktivitätsdiagramm und zu Arena die Hypothese als wahr bestätigt wird.

Neben diesen drei Hypothesen, die für die einzelnen Quell- und Zielmodelle betrachtet werden müssen, sind noch vierte und fünfte Hypothese, im Kontext der vierte Zielstellung der Arbeit, zu betrachten:

H4 Wenn die Bezeichnungen der Elemente der auszuführenden Tätigkeiten Informationen über den Geschäftsprozess beinhalten, dann können diese für die Beschaffung von simulationsrelevanten Informationen verwendet werden.

H5 Wenn die Bezeichnungen der Elemente der auszuführenden Tätigkeiten in einem Geschäftsprozessmodell hinsichtlich ihrer Wortart betrachtet werden, dann können daraus Informationen für die Simulation eines Geschäftsprozessmodells abgeleitet werden.

Beide Hypothesen können als wahr angesehen werden. Der Beleg liegt im Repository als auch in den linguistischen Normalisierungsregeln. Im Kontext der vierten Hypothese wird auf die Methode zur Festlegung der Bearbeitungszeit verwiesen. Diese nutzt die Bezeichnungen beziehungsweise die Verben für die Festlegung. Aber auch linguistische Normalisierungsregeln nutzen die Bezeichnung. Insbesondere sind hier die mit dem Repository im Zusammenhang stehenden Normalisierungsregeln aNR_{13} und aNR_{14} anzubringen. Während die vierte Hypothese umfassender betrachtet wurde, ist dies bei der Fünften nur in Ansätzen der Fall. Dennoch kann diese als wahr, durch die linguistische Normalisierungsregel aNR_{15}, angesehen werden. Die Bedeutung der Wortarten und des Kasus bedarf aber noch weitere Untersuchungen, damit die indirekt verwendeten Informationen in den Bezeichnungen weiterführend verwendet werden können.

5.2 Resultierende Forschungsfragen

Als abschließenden Punkt erfolgt eine kompakte Darlegung der offenen Forschungsfragen, die sich im Kontext der Arbeit ergeben haben.

Die Transformationsregeln beziehen sich jeweils nur auf Notationen zur Überführung von Geschäftsprozessmodellen in das ProSiT Transformationsmodell. Wird aber ein Organigramm betrachtet, dann könnte dieses in die Ressourcensicht des ProSiT Konzepts überführt werden. So kann über die Anzahl der Personen, die einer Stelle zugeordnet sind, die Anzahl der zur Verfügung stehenden Ressourcen automatisch bestimmt werden. Die Transformationsregeln die einen BPMN Prozess überführen, können, bezogen auf die Aufgabe, detaillierter untersucht werden. Eine Aufgabe kann in BPMN mit einem Aufgabentyp versehen werden. Dieser wird im ProSiT Ablaufdiagramm aber nicht berücksichtigt. Insbesondere auf die Erfassung der Bearbeitungszeit könnte der Aufgabentyp verwendet werden, eine passende Methode auszuwählen. Ob der Aufgabentyp weiterführend für eine Simulationsstudie verwendet werden kann, ist darüber hinaus allgemein zu untersuchen.

Die Normalisierung ist zum einen mit weiteren Regeln anzureichern, welche eine Vorbereitung eines Geschäftsprozessmodells für eine Simulation ermöglichen, zum anderen sollten in Anlehnung an Staud (2005, S. 48) Normalisierungsformen festgelegt werden. Zum einen können diese als Zwischenschritte definiert werden, beziehungsweise für die Konzeption eines Vorgehensmodells der Anwendung der Normalisierung verwendet werden, zum anderen fehlt in der aktuellen Konzeption eine Spezifikation, wann die Normalisierung abgeschlossen ist. Für diese Spezifikation sind Kriterien zu definieren, die erfüllt sein müssen, damit eine Überführung in eine Simulationsumgebung erfolgen kann. Ebenfalls ist ein Ansatz zu entwickeln, mit denen Normalisierungsregeln objektiv geprüft werden können, insbesondere hinsichtlich des semantischen Verhaltens.

Ein großes Forschungsfeld sind Ressourcen beziehungsweise eine detailliertere Konzeption eines Ressourcenmodells für die Simulation von Geschäftsprozessen, welches jedoch auch von den Zielsimulationsumgebungen unterstützt werden muss. Hierfür sollte das Ressourcensystem des Werkers in Plant Simulation für die Verwendung in Betracht gezogen werden, bei dem ein Werker mehrere Dienste anbieten. Die Dienste selbst werden beispielsweise von Einzelstationen angefordert, bei dem jeder Werker, der diesen Dienst anbietet, die Arbeit erledigen kann. Bezogen auf eine Aktivität im ProSiT Ablaufdiagramm, könnte diese, anstelle einer konkreten Ressource, einen Dienst oder eine Kompetenz anfordern, der notwendig ist, damit die Aktivität ausgeführt werden kann. Ein Ansatz der Kompetenzen in Geschäftsprozessmodellen berücksichtigt, kann beispielsweise bei Leyking und Angeli (2008) gefunden werden. Bezogen auf die Ausführung einer Aktivität könnte eine andere Ressource einspringen, wenn diese die Kompetenz aufweist, aber diese Tätigkeit für gewöhnlich nicht ausführt, womit es, wie in der Realität, nicht zu einer Aufstauung von Arbeitspaketen kommt, wodurch nachfolgende Bearbeiter die eigenen Aufgaben nicht mehr ausführen können.

In diesem Zusammenhang ist zu untersuchen, inwieweit sich einzelne Ressourcen auf die Bearbeitungszeit auswirken oder wie Abarbeitungsstrategien von Aufgaben unterstützt werden können, wie diese van der Aalst et al. (2010, S. 320) aufführt. Wechselt eine Marke von einer stationären Ressource zu einer anderen, dann ist dies ein Anzeichen, dass eine Wegstrecke; zum einen für die Marke, zum anderen für die anderen Ressourcen vorliegt. Hierfür ist ein Konzept zu entwickelt, wie dies im Rahmen des ProSiT Ablaufdiagramms und in den Simulationsmodellen berücksichtigt werden kann, wobei auch Kriterien zu definieren sind, wann eine Anwendung des Konzepts notwendig ist.

Eine Mischung zwischen Ressourcen und Transformationsregeln liegt bei Informationssystemen vor. In der aktuellen Konzeption wird unterstellt, dass diese nicht simulationsrelevant sind. Da aber auch Geschäftsprozesse simuliert werden können, die teilweise oder komplett in Informationssystemen umgesetzt werden, ist zu prüfen inwieweit diese als Ressourcen berücksichtigt werden können und wie diese aus Quellmodellen in das Transformationsmodell überführt werden können. Ein erster Ansatz hierfür kann in Schmietendorf und End (2009) gefunden werden, der für das ProSiT Konzept adaptiert werden kann.

Neben der Ressourcensicht ist auch die Tätigkeitssicht beziehungsweise das Repository weiter zu entwickeln. Die Normalisierungsregel aNR_{16} prüft beispielsweise eine Tätigkeit nach dem Vorhandensein des Wortes „überwachen". Neben diesem Wort sind aber noch weitere Worte wie beispielsweise „unterstützten oder „assistieren" denkbar, die auf eine zeitabhängige Aktivität hinweisen. So könnte das Repository um entsprechende Tabellen erweitert werden, damit mit diesen zeitabhängige Aktivitäten identifiziert werden können.

Bezogen auf Workflows entwickelten van der Aalst er al. (2003) Workflow Patterns, mit denen die Leistungsfähigkeit einer Notation für die Unterstützung von Workflows untersucht werden kann. Konzepte, die in den Workflow Patterns unterstützt werden, könnten auf das ProSiT Konzept überführt werden, um dessen Leistungsfähigkeit zu erhöhen. Da mit dem ProSiT Konzept aber keine Workflows im Vordergrund stehen, sondern die Simulation adressiert wird, stellt sich ein interessanter Forschungspunkt dar, die Workflow Patterns auf die Simulation von Geschäftsprozessen zu überführen. Als offener Forschungspunkt kann daher die Konzeption von Business Process Simulation Patterns angeführt werden. Mit diesen könnten Zielumgebungen hinsichtlich ihrer Eignung für die Simulation von Geschäftsprozessen überprüft werden.

Bezogen auf weitere Zielumgebungen stehen zwei Simulationsumgebungen im Vordergrund, zum einen Plant Simulation, zum anderen DESMO-J. Durch die Vorarbeiten für die Transformationsregeln, insbesondere durch die Arbeit von Just (2010), liegt der Schwerpunkt der weiteren Entwicklung in der Umsetzung eines Ressourcenmodells, welches die Ressourcenbindung als auch die Ressourcenfreigabe unterstützt. Ebenfalls ist eine Lösungsansatz für Synchronisierung in Plant Simulation zu entwickelt, mit dem nicht beliebige Marken miteinander verschmolzen werden, sondern Marken der gleichen Instanz. Dieses Verhalten muss auch noch für die Synchronisation in AnyLogic umgesetzt werden, was jedoch der Attributsebene zuzuordnen ist. Erste Untersuchungen zu DESMO-J zeigen, dass dies geeignet ist, das ProSiT Ablaufdiagramm direkt abzubilden. Da es sich aber, mit Bezug auf die Java Klassen, um eine Simulationssprache handelt stellt diese Zielumgebung eine Abweichung zu den anderen Systemen dar, bei denen nicht direkt mit der Simulationssprache das Simulationsmodell erstellt wurde. Hierbei wird die These aufgestellt, dass mit Simulationssprachen einfacher das ProSiT Ablaufdiagramm simuliert werden kann, als wenn dieses in Simulationsumgebungen überführt wird, die das Simulationsmodell über ein grafisches Modell erzeugen.

Die sprachliche Analyse von Geschäftsprozessmodellen stellt einen weiteren Forschungspunkt dar. Die linguistischen Normalisierungsregeln stellen hierfür einen ersten Ansatz dar, sprachliche Aspekte für weiterführende Zwecke zu verwenden. Ansatzweise wurde dieser Aspekt bei der automatischen Normalisierungsregel aNR_{15} aufgeführt. Denkbar wäre ein auf Wortarten basierender Ansatz, der in der Lage ist, aus einem Fließtext die Rohfassung eines Geschäftsprozessmodells automatisch zu erzeugen.

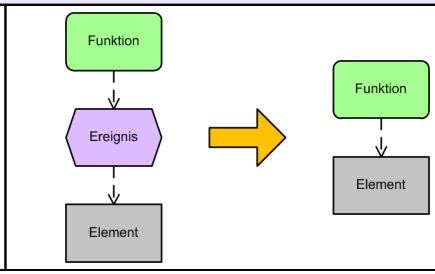
Da über linguistische Normalisierungsregeln die Klassifikation einer Aktivität in die Unterstützungs- und Hauptaktivität erfolgt, stellt sich in diesem Zusammenhang die Forschungsfrage, inwieweit diese Unterscheidung im Rahmen einer Simulationsstudie verwendet werden kann, um Verbesserungsmöglichkeiten und Schwachstellen eines Geschäftsprozesses ausfindig zu machen. Ebenfalls bietet diese Unterscheidung auch einen Ansatzpunkt, die Granularität eines Geschäftsprozessmodells zu bestimmen. Im Hinblick auf die Simulationswürdigkeit könnte dieser Ansatz verwendet werden, um zu bestimmen, ob ein Geschäftsprozess eine angemessene Granularität aufweist. Bezogen auf die Simulationswürdigkeit wurde auch der Ansatz von Mendling (2008) zur Fehleranfälligkeit von eEPK Modell aufgeführt. Eine Weiterentwicklung dieses kann ebenfalls herangezogen werden, die Simulationswürdigkeit eines Geschäftsprozessmodells objektiv zu bewerten.

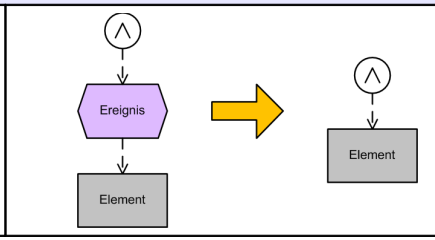
Als allgemeine Forschungspunkte kann das Hinzufügen weiterer Notationen zur Modellierung von Geschäftsprozessen sowie Simulationsumgebungen und die Weiterentwicklung der Normalisierungsregeln aufgeführt werden. Das ProSiT Konzept bietet hierfür die Grundlage, mit der die Überführung und Vorbereitung von Geschäftsprozessen für ein Simulationsvorhaben systematisch unterstützt werden kann.

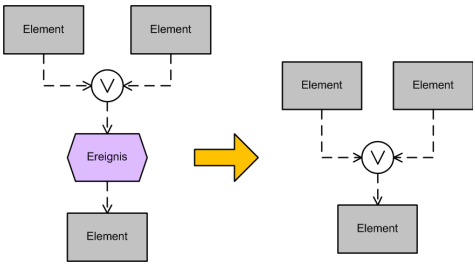
Anhang A: Regelbasis

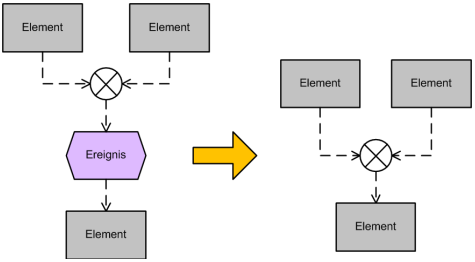
A.1 Regelbasis der eEPK

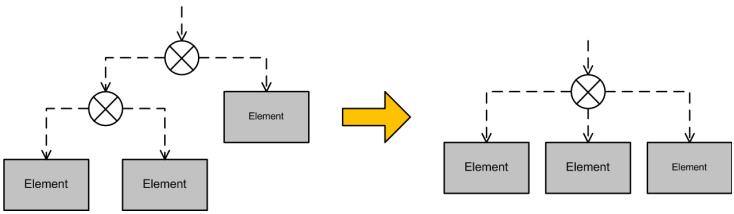
A.1.1 eEPK Vorbereitungsregeln

Preparation Rule ePR_1	
If an event has a function as its predecessor and it has a successor, then it is removed.	

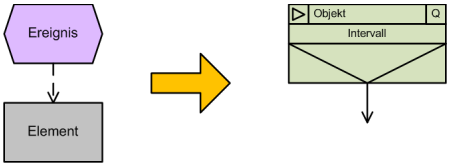
Preparation Rule ePR_2	
If an event has an AND operator as its predecessor and it has a successor, then it is removed.	

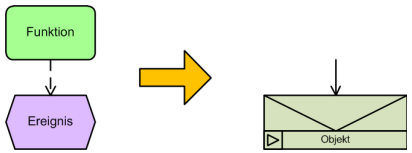
Preparation Rule ePR_3	
If an event has an OR operator as its predecessor and it has a successor and the OR operator has more than one predecessor, then it is removed.	

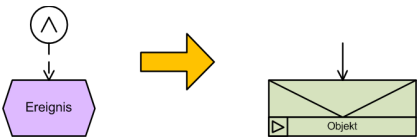
Preparation Rule ePR_4	
If an event has a XOR operator as its predecessor and it has a successor and the XOR operator has more than one predecessor, then it is removed.	

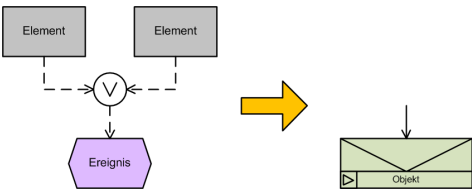
Preparation Rule ePR_5
If a XOR operator has more than one successors and a XOR operator, with more than one successors, as its predecessor, the XOR Operator is removed and all successors will become successors of its predecessor XOR.


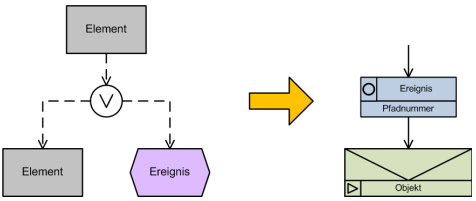
A.1.2 eEPK Transformationsregeln

Transformation Rule eTR_1	
If an event has not a predecessor, then it will be a source.	

Transformation Rule eTR_2	
If an event has a function as its predecessor and it has not a successor, then it will be a sink.	

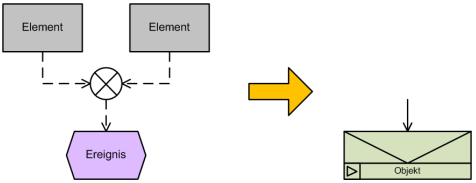
Transformation Rule eTR_3	
If an event has an AND operator as its predecessor and it has not a successor, then it will be a sink.	

Transformation Rule eTR_4	
If an event has an OR operator as its predecessor and it has not a successor and the OR operator has more than one predecessor, then it will be a sink.	

Transformation Rule eTR_5	
If an event has an OR operator as its predecessor and it has not a successor and the OR operator has one predecessor, then it will be an inclusive key and a sink, that is the successor of the key.	

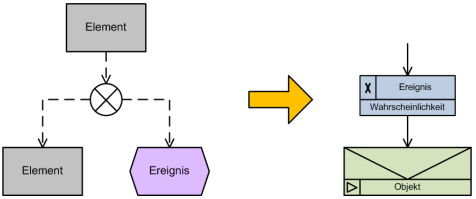
Transformation Rule eTR_6

If an event has a XOR operator as its predecessor and it has not a successor and the XOR operator has more than one predecessor, then it will be a sink.



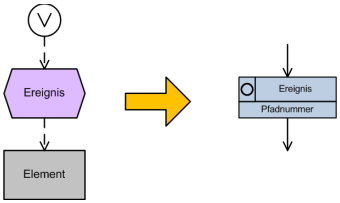
Transformation Rule eTR_7

If an event has a XOR operator as its predecessor and it has not a successor and the XOR operator has one predecessor, then it will be an exclusive key and a sink, that is the successor of the key.



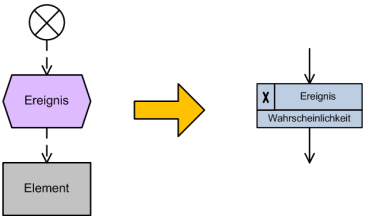
Transformation Rule eTR_8

If an event has an OR operator as its predecessor and it has a successor, then it will be an inclusive key.



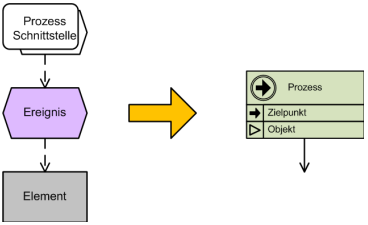
Transformation Rule eTR_9

If an event has a XOR operator as its predecessor and it has a successor, then it will be an exclusive key.



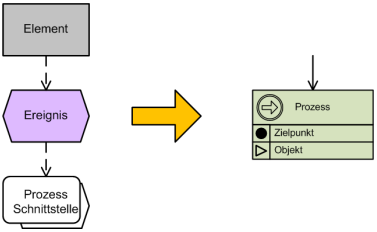
Transformation Rule eTR_{10}

If an event has a process interface as its predecessor and it has a successor, then the event and the process interface will be a target point.



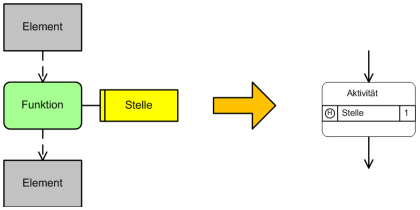
Transformation Rule eTR_{11}

If an event has a process interface as its successor and it has a predecessor, then the event and the process interface will be a jump point.



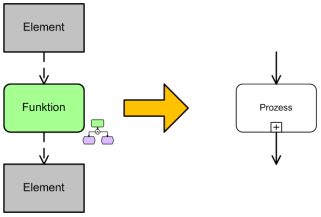
Transformation Rule eTR_{12}

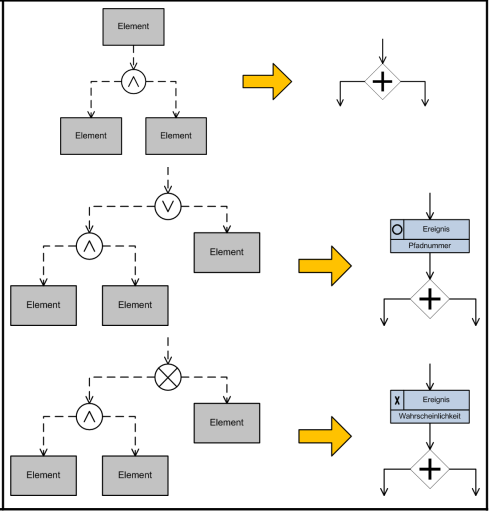
If a function has not a linked process, then it will be an activity. If participants are related to the function, they will be added as a human resource to the activity.

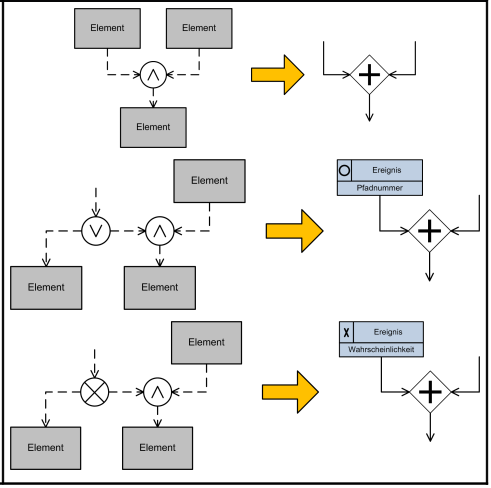


Transformation Rule eTR_{13}

If a function has a linked process, then it will be sub-process.

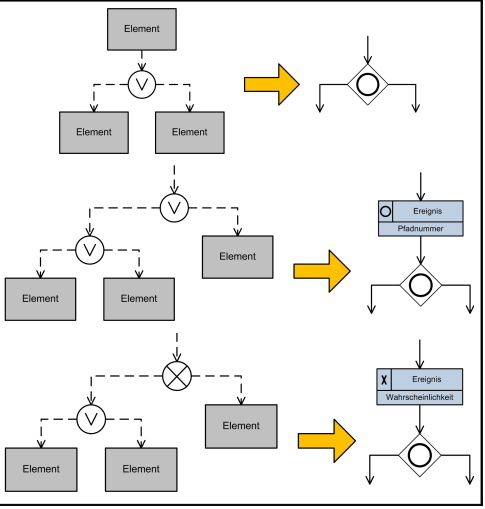


Transformation Rule eTR_{14}	
If an AND operator has one predecessor and more than one successor, then it will be a parallel split gateway and if the predecessor is an OR operator that has more than one successor, it will get an inclusive key as predecessor of the gateway and if the predecessor is a XOR operator that has more than one successor, it will get an exclusive key as predecessor of the gateway.	

Transformation Rule eTR_{15}	
If an AND operator has more than one predecessor and one successor, then it will be a parallel join gateway and every incoming flow that has an OR operator as predecessor, that has more than one successor will get an inclusive key as predecessor of the gateway and every incoming flow that has a XOR operator as predecessor, that has more than one successor, will get an exclusive key as predecessor of the gateway.	

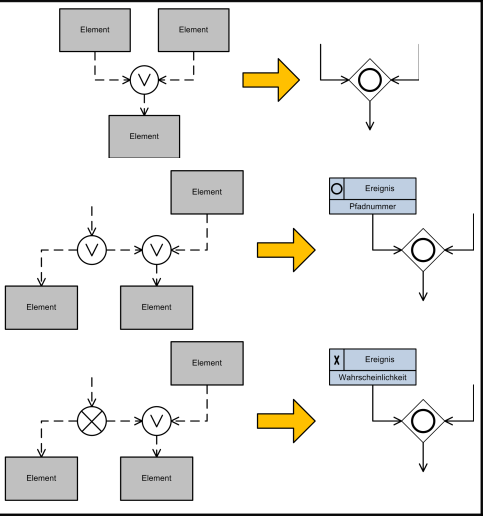
Transformation Rule eTR$_{16}$

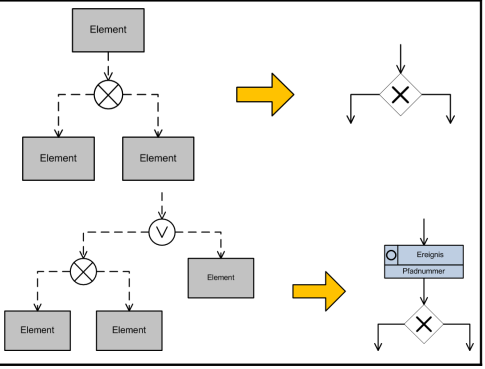
If an OR operator has one predecessor and more than one successor, then it will be an inclusive split gateway and
if the predecessor is an OR operator as predecessor that has more than one successor, it will get an inclusive key as predecessor of the gateway and
if the predecessor is a XOR operator as predecessor that has more than one successor, it will get an exclusive key as predecessor of the gateway.

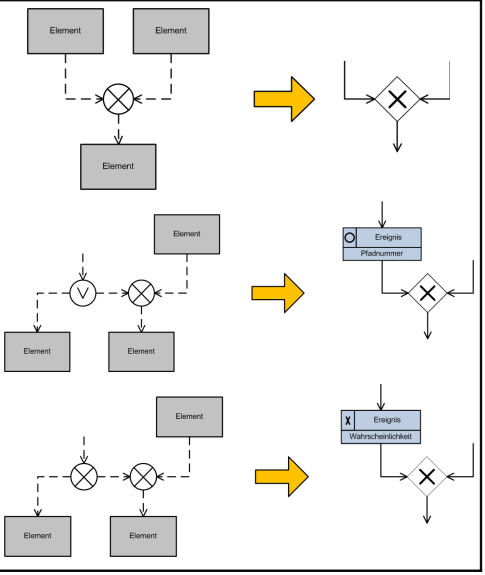


Transformation Rule eTR$_{17}$

If an OR operator has more than one predecessor and one successor, then it will be an inclusive join gateway and
every incoming flow that has an OR operator as predecessor, that has more than one successor will get an inclusive key as predecessor of the gateway and
every incoming flow that has a XOR operator as predecessor, that has more than one successor, will get an exclusive key as predecessor of the gateway.

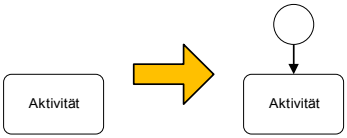


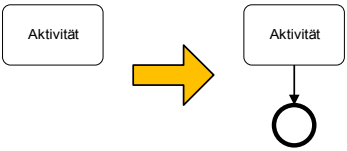
Transformation Rule eTR_{18}	
If a XOR operator has one predecessor and more than one successor, then it will be an exclusive split gateway and if the predecessor is an OR operator as predecessor that has more than one successor, it will get an inclusive key as predecessor of the gateway.	

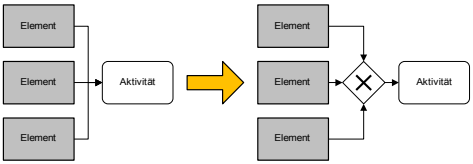
Transformation Rule eTR_{19}	
If a XOR operator has more than one predecessor and one successor, then it will be an exclusive join gateway and every incoming flow that has an OR operator as predecessor, that has more than one successor will get an inclusive key as predecessor of the gateway and every incoming flow that has a XOR operator as predecessor, that has more than one successor, will get an exclusive key as predecessor of the gateway.	

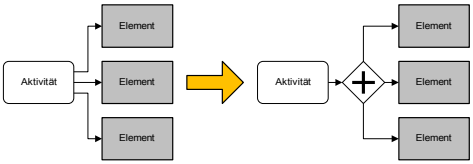
A.2 Regelbasis von BPMN

A.2.1 BPMN Vorbereitungsregeln

Preparation Rule bPR_{1_1}	
If an activity has not a predecessor and it has one or more successors, then it will get a start event as predecessor.	

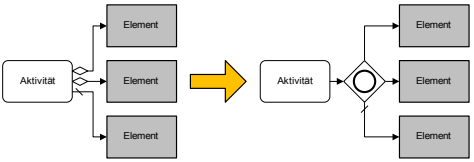
Preparation Rule bPR_{1_2}	
If an activity has one one more predecessors and it has not a successor, then the activity will get an end event as successor.	

Preparation Rule bPR_{1_3}	
If an activity has more than one predecessor, then the activity will get an exclusive gateway as predecessor and all predecessors of the activity will get the predecessors of the gateway.	

Preparation Rule bPR_{1_4}	
If an activity has more than one successor, then the activity will get a parallel gateway as successor and all successors of the activity will get the successors of the gateway.	

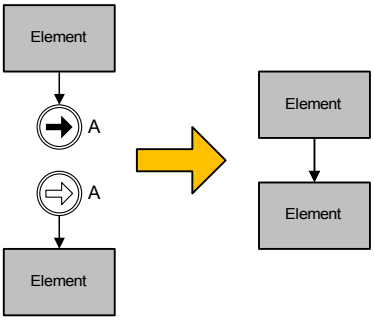
Preparation Rule bPR_{1_5}

If an activity has one or more conditional flow or a default flow for the connection to its successors, then the conditional flows are replaced by a sequence flow and the activity will get an inclusive gateway as successor and all successors of the activity will become the successors of the gateway.



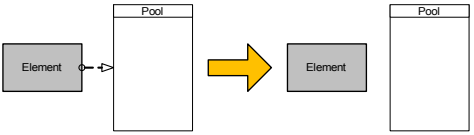
Preparation Rule bPR_{1_6}

If an intermediate link throwing event has the same labeling as an intermediate link catching event, then both elements are removed and the predecessor of the intermediate link throwing event is connected to the successor of the intermediate link catching event.



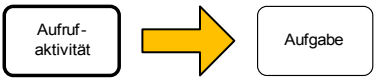
Preparation Rule bPR_{1_7}

If an element has an outgoing message flow that is connected to a pool, then the message flow is removed.



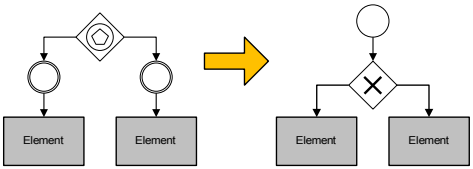
Preparation Rule bPR_{1_8}

If the element is a call activity, then it will be replaced by the linked element.



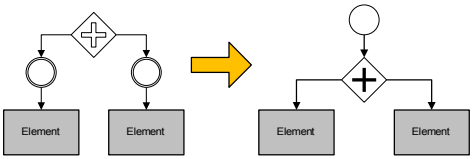
Preparation Rule bPR_{1_9}

If the element is an exclusive event-based gateway, then the gateway is replaced by an exclusive gateway and the successor intermediate events are removed and the exclusive gateway will get a start event as predecessor.



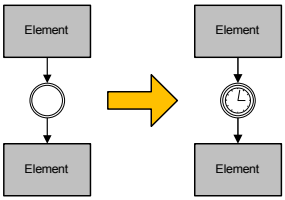
Preparation Rule bPR_{1_10}

If the element is a parallel event-based gateway, then the gateway is replaced by a parallel gateway and the successor intermediate events are removed and the parallel gateway will get a start event as predecessor.



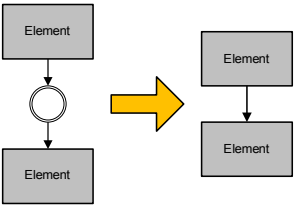
Preparation Rule bPR_{1_11}

If the element is a catching intermediate event and it has not an incoming message flow, then it will be replaced by an intermediate timer event.



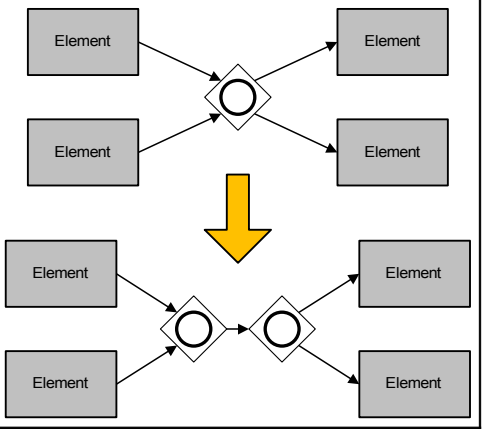
Preparation Rule bPR_{1_12}

If the element is a throwing intermediate event and it has not an outgoing message flow, then it will be removed.



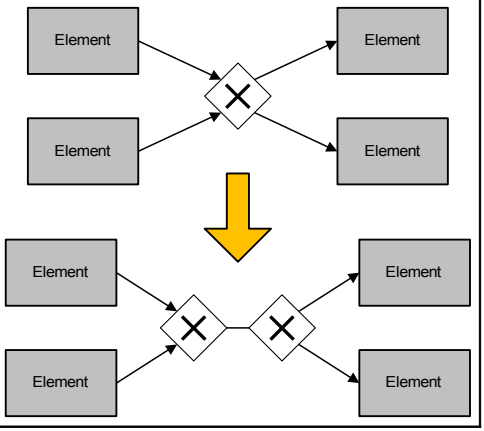
Preparation Rule bPR_{1_13}

If an inclusive gateway has more than one predecessor and more than one successor, than the gateway will get an additional inclusive gateway as successor and every successor of the gateway will become successors of the additional gateway.



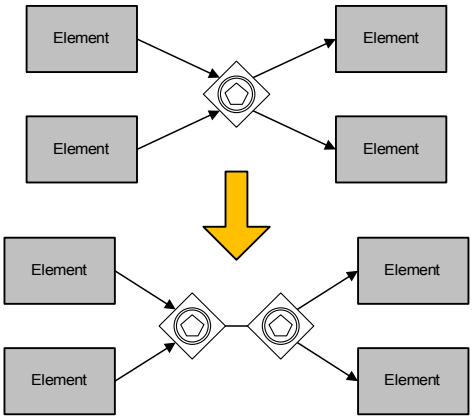
Preparation Rule bPR_{1_14}

If an exclusive gateway has more than one predecessor and more than one successor, than the gateway will get an additional exclusive gateway as successor and every successor of the gateway will become successors of the additional gateway.



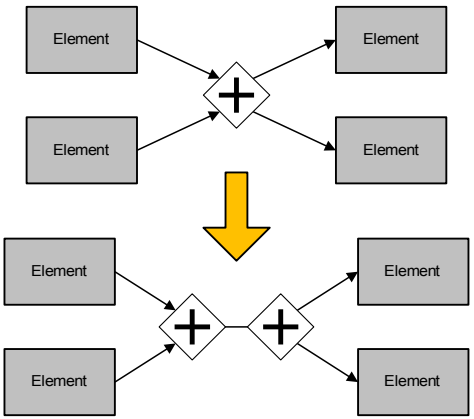
Preparation Rule bPR_{1_15}

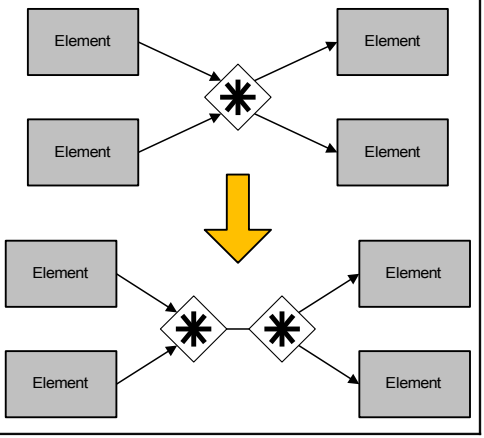
If an event-based gateway has more than one predecessor and more than one successor, than the gateway will get an additional event-based gateway as successor and every successor of the gateway will become successors of the additional gateway.



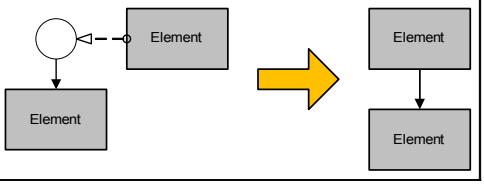
Preparation Rule bPR_{1_16}

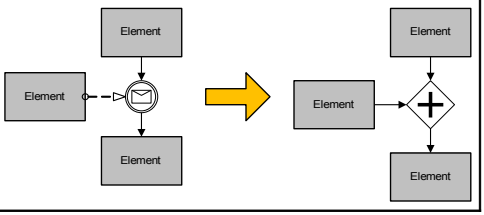
If a parallel gateway has more than one predecessor and more than one successor, than the gateway will get an additional parallel gateway as successor and every successor of the gateway will become successors of the additional gateway.

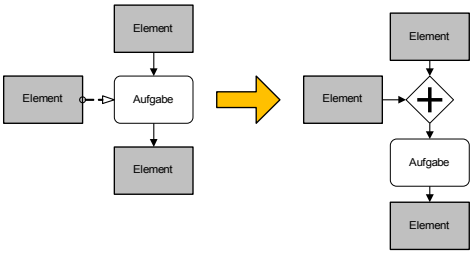


Preparation Rule bPR_{1_17}	
If a complex gateway has more than one predecessor and more than one successor, than the gateway will get an additional complex gateway as successor and every successor of the gateway will become successors of the additional gateway.	

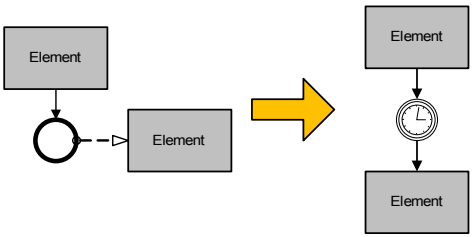
Vorbereitungsregelserie bPR_{N_x}

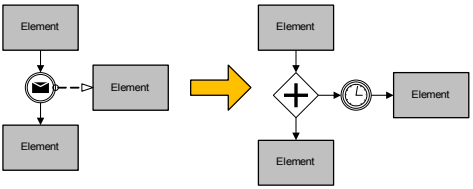
Preparation Rule bPR_{N_1}	
If a start event has an incoming message flow, then it will be removed.	

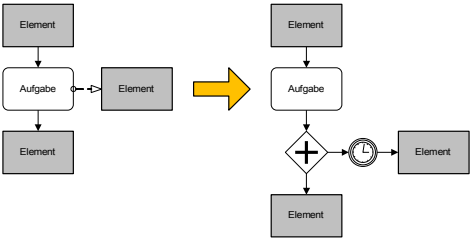
Preparation Rule bPR_{N_2}	
If an intermediate message catching event has an incoming message flow, then it will be converted to a parallel gateway.	

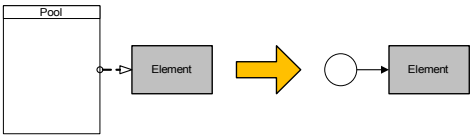
Preparation Rule bPR_{N_3}	
If a task has an incoming message flow, then the task will get a parallel gateway as predecessor and the element that is connected with the message flow will get a predecessor of the gateway.	

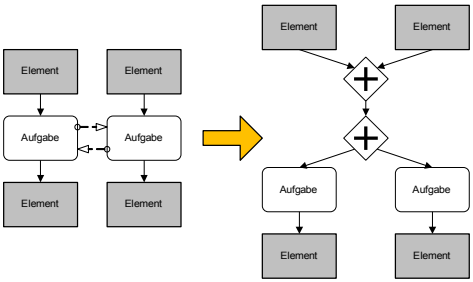
Vorbereitungsregelserie bPR_{2_x}

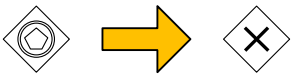
Preparation Rule bPR_{2_1}	
If an end event has an outgoing message flow, then it will be converted to a timer intermediate event and the message flow is converted to a sequence flow and if the target of the message flow is a start event, then the rule bPR_{N_1} is executed or if the target is an intermediate event, then the rule bPR_{N_2} is executed or if the target is a task, then the rule bPR_{N_3} is executed.	

Preparation Rule bPR_{2_2}	
If an intermediate message throwing event has an outgoing message flow, then it will be converted to a parallel gateway and the gateway will get a timer event as a successor and this event will get the target of the message flow as successor and if the target of the message flow is a start event, then the rule bPR_{N_1} is executed or if the target is an intermediate event, then the rule bPR_{N_2} is executed or if the target is a task, then the rule bPR_{N_3} is executed.	

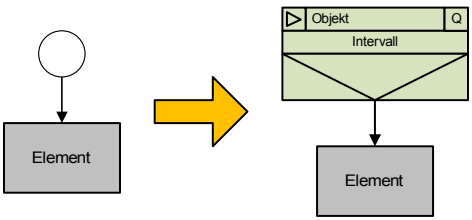
Preparation Rule bPR_{2_3}	
If a task has an outgoing message flow and not an incoming message flow from the same element, then the task will get a parallel gateway as successor and the gateway will get a timer event as a successor and this event will get the target of the message flow as successor and if the target of the message flow is a start event, then the rule bPR_{N_1} is executed or if the target is an intermediate event, then the rule bPR_{N_2} is executed or if the target is a task, then the rule bPR_{N_3} is executed.	

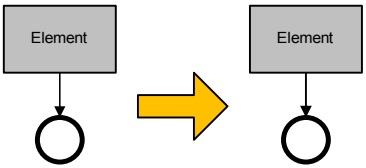
Preparation Rule bPR_{2_4}	
If a pool has an outgoing message flow, then a source will become the source of the message flow and if the target of the message flow is a start event, then the rule bPR_{N_1} is executed or if the target is an intermediate event, then the rule bPR_{N_2} is executed or if the target is a task, then the rule bPR_{N_3} is executed.	

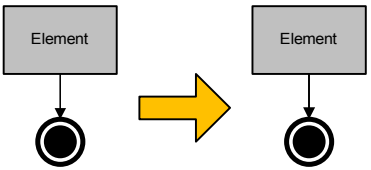
Preparation Rule bPR_{2_5}	
If a task has an outgoing and incoming message flow to an other task, then both tasks will get a parallel gateway as predecessor and this gateway will get a parallel gateway as predecessor that will get the predecessors of the tasks as its predecessors.	

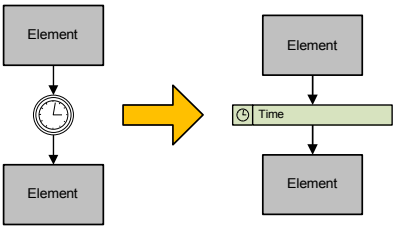
Preparation Rule bPR_{2_6}	
If the element is an event-based gateway, then it will be an exclusive gateway.	

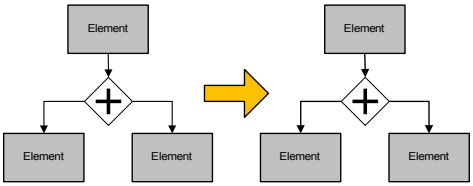
A.2.2 BPMN Transformationsregeln

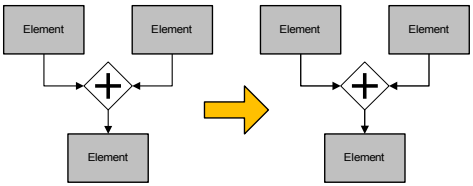
Transformation Rule bTR_1	
If the element is a start event, then it will be a source	

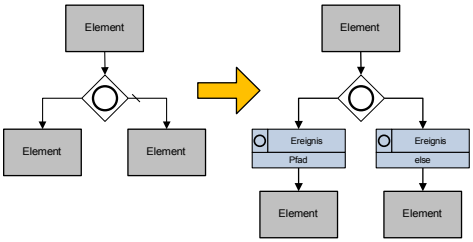
Transformation Rule bTR_2	
If the element is an end event and it has not a terminate marker, then it will be a sink.	

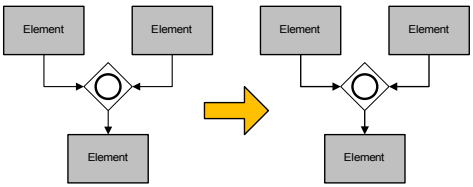
Transformation Rule bTR_3	
If the element is a terminate end event, then it will be a terminating sink.	

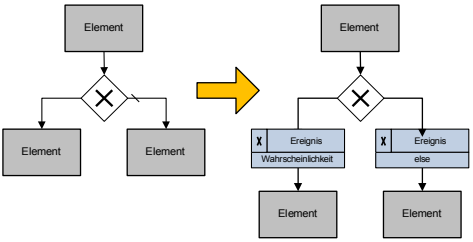
Transformation Rule bTR_4	
If the element is an intermediate timer event, then it will be a delay.	

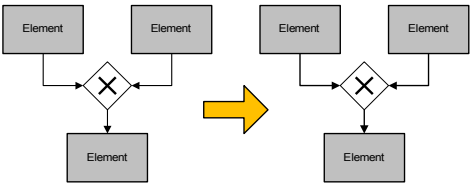
Transformation Rule bTR_5	
If a parallel gateway has more than one successor, then it will be a parallel split gateway.	

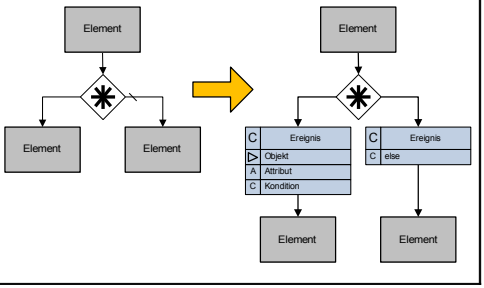
Transformation Rule bTR_6	
If a parallel gateway has more than one predecessor, then it will be a parallel join gateway.	

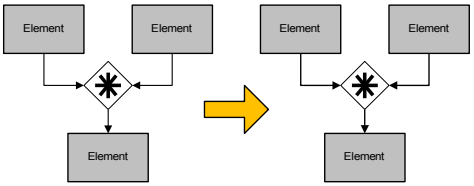
Transformation Rule bTR_7	
If an inclusive gateway has more than one successor, then it will be an inclusive split gateway and every successor of the gateway will get an inclusive key as predecessor that is the successor of the split gateway and if the inclusive gateway has a standard flow, then the corresponding inclusive key will be marked with an else.	

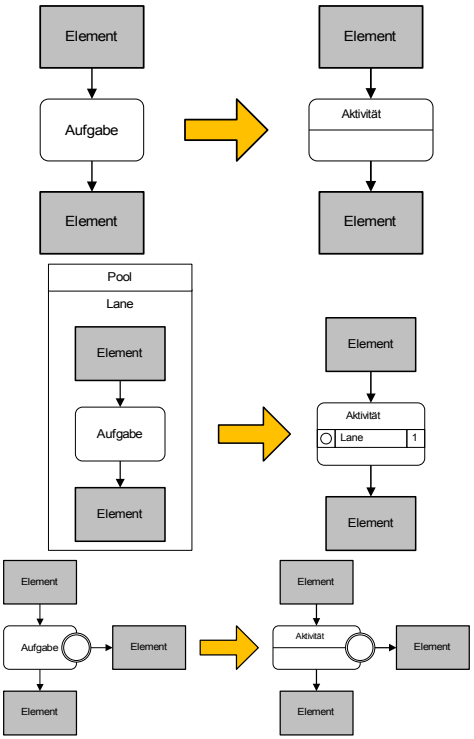
Transformation Rule bTR_8	
If an inclusive gateway has more than one predecessor, then it will be an inclusive join gateway.	

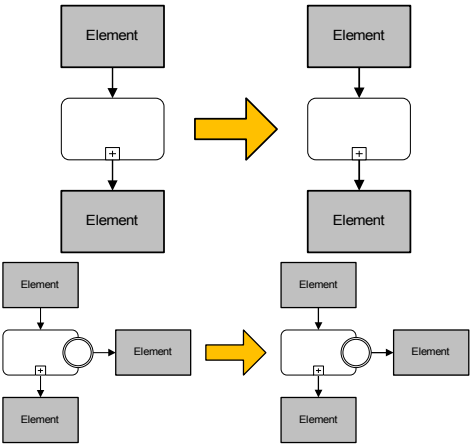
Transformation Rule bTR_9	
If an exclusive gateway has more than one successor, then it will be an exclusive split gateway and every successor of the gateway will get an exclusive key as predecessor that is the successor of the split gateway and if the exclusive gateway has a standard flow, then the corresponding exclusive key will be marked with an else.	

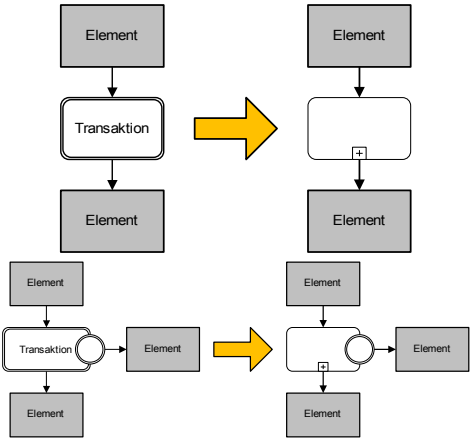
Transformation Rule bTR_{10}	
If an exclusive gateway has more than one predecessor, then it will be an exclusive join gateway.	

Transformation Rule bTR_{11}	
If a complex gateway has more than one successor, then it will be a complex split gateway and every successor of the gateway will get an exclusive key as predecessor that is the successor of the split gateway and if the complex gateway has a standard flow, then the corresponding complex key will be marked with an else.	

Transformation Rule bTR_{12}	
If a complex gateway has more than one predecessor, then it will be a complex join gateway.	

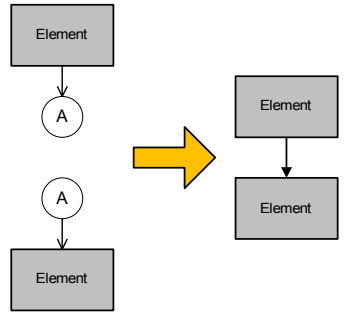
Transformation Rule bTR_{13}	
If the element is a task, then it will be an activity and if the task is placed in a lane, then the deepest lane is added as a resource and if the task has boundary events, these events will be added to the activity.	

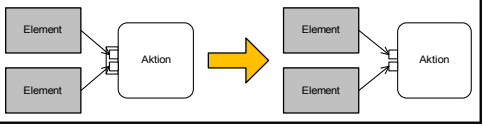
Transformation Rule bTR_{14}	
If the element is a sub process, then it will be a sub process and if the sub process has boundary events, these events will be added to the sub process.	

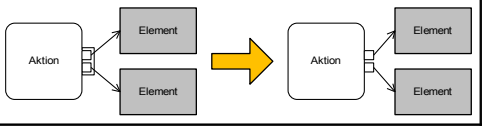
Transformation Rule bTR_{15}	
If the element is a transaction, then it will be a sub process and if the transaction has boundary events, these events will be added to the sub process.	

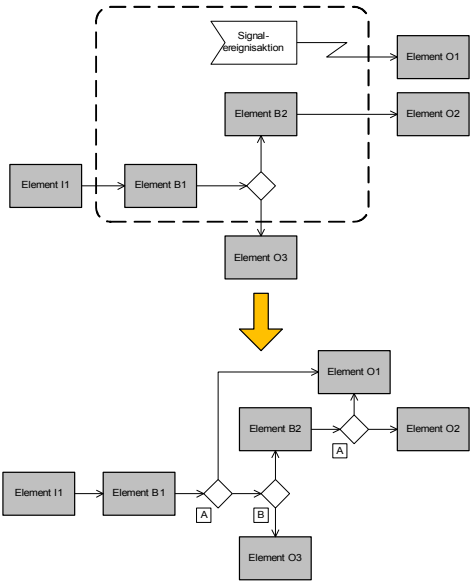
A.3 Regelbasis des UML Aktivitätsdiagramms

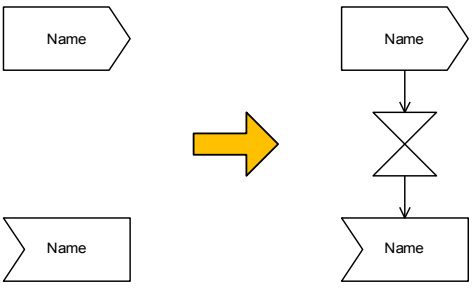
A.3.1 UML AD Vorbereitungsregeln

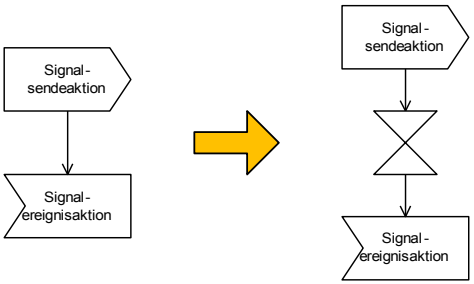
Preparation Rule adPR$_{1_1}$	
If an outgoing connector and an incoming connector have the same name, then both elements are removed and the predecessor of the outgoing connector will become the predecessor of the successor of the incoming connector.	

Preparation Rule adPR$_{1_2}$	
If an action has a parameter set with incoming process flows, then the parameter set is removed.	

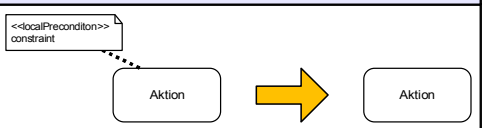
Preparation Rule adPR$_{1_3}$	
If an action has a parameter set with outgoing process flows, then the parameter set is removed.	

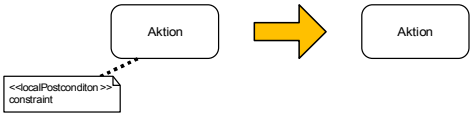
Preparation Rule adPR$_{1_4}$	
If an expansion region contains a signal event action that has not a predecessor and an outgoing exception handler, then the expansion region and the signal event action are removed and all elements that are not a decision node will get a decision node as successor that has a first successor the successor of the element and as a second successor the element that is linked with the exception handler.	

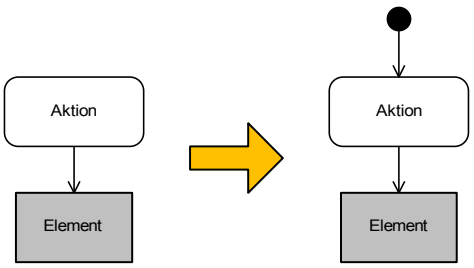
Preparation Rule adPR$_{1_5}$	
If an accept event action has the same name as a send signal action, than a new time event is added to the model and the send signal action will get the time event as successor and the accept event action will get the time event as predecessor.	

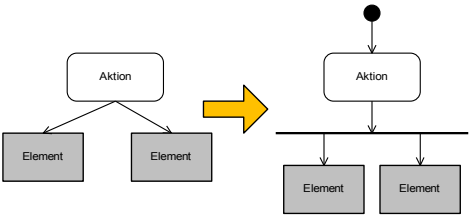
Preparation Rule $adPR_{1\_6}$	
If a signal send action has a signal event action as a predecessor and the model does not contain a signal event action with the same name as the signal send action, than the connection between those two elements is removed and a new time event is added to the model and the send signal action will get the time event as its successor and the accept event action will get the time event as its predecessor.	

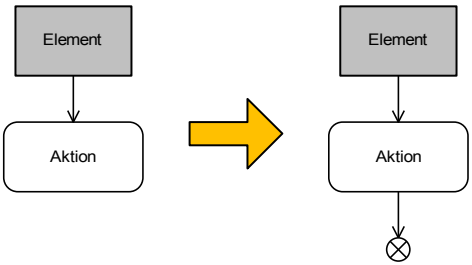
Vorbereitungsregelserie $adPR_{2\_x}$

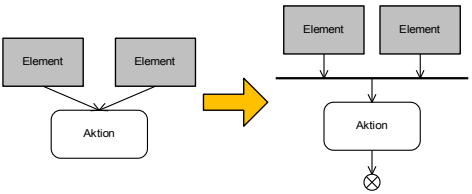
Preparation Rule $adPR_{2\_1}$	
If an action has local preconditions, then the local preconditions are removed.	

Preparation Rule $adPR_{2\_2}$	
If an action has local post conditions, then the local post conditions are removed.	

Preparation Rule $adPR_{2\_3}$	
If the element is an action or an accept event action or a send signal action or a call action or an expansion region and the element has not a predecessor and one successor, then the element will get an initial node as its predecessor.	

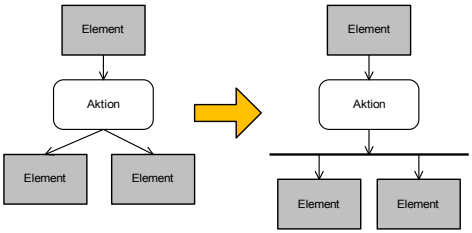
Preparation Rule adPR$_{2_4}$	
If the element is an action or an accept event action or a send signal action or a call action or an expansion region and the element has not a predecessor and more than one successor, then the element will get an initial node as its predecessor, a fork node as a successor and every successor of the element will become the successor of the fork node.	

Preparation Rule adPR$_{2_5}$	
If the element is an action or an accept event action or a send signal action or a call action or an expansion region and the element has not a successor and one predecessor, then the element will get a flow final as its successor.	

Preparation Rule adPR$_{2_6}$	
If the element is an action or an accept event action or a send signal action or a call action or an expansion region and the element has not a successor and more than one predecessor, then the element will get a flow final as its successor, a join node as a predecessor and every predecessor of the element will become the predecessor of the fork node.	

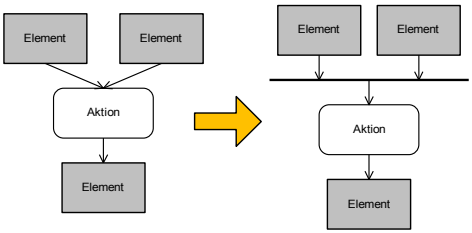
Preparation Rule adPR$_{2_7}$

If the element is an action or an accept event action or a send signal action or a call action or a time event or an expansion region and the element has a predecessor and more than one successor,
then the element will get a fork node as a successor and
every successor of the element will become the successor of the fork node.



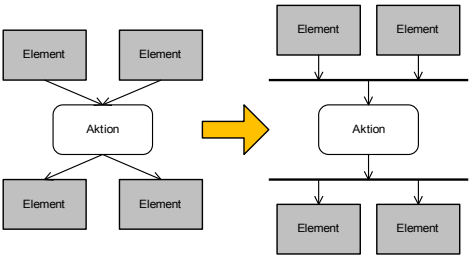
Preparation Rule adPR$_{2_8}$

If the element is an action or an accept event action or a send signal action or a call action or a time event or an expansion region and the element more than one predecessor and a successor,
then the element will get a join node as a predecessor and
every predecessor of the element will become the predecessor of the join node.



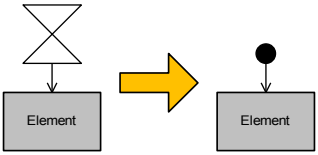
Preparation Rule adPR$_{2_9}$

If the element is an action or an accept event action or a send signal action or a call action, or a time event or an expansion region and the element has more than one predecessor and more than one successor,
then the element will get a join node as a predecessor,
every predecessor of the element will become the predecessor of the join node and
the element will get a fork node as a successor and
every successor of the element will become the successor of the fork node.



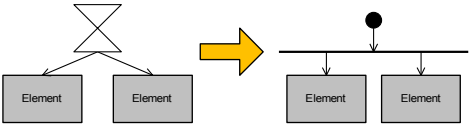
Preparation Rule adPR$_{2_10}$

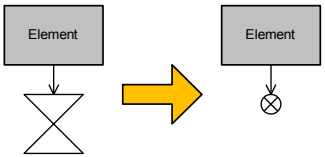
If a time event has not a predecessor and it has a successor, then the time event is removed and the successor will get an initial node as its predecessor.

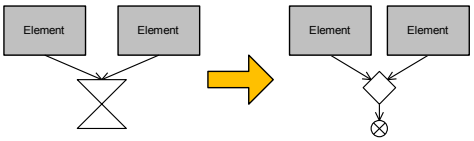


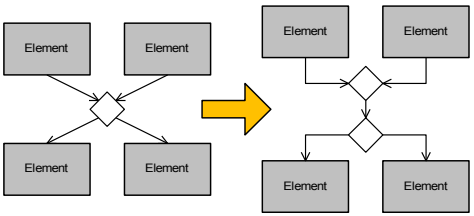
Preparation Rule adPR$_{2_11}$

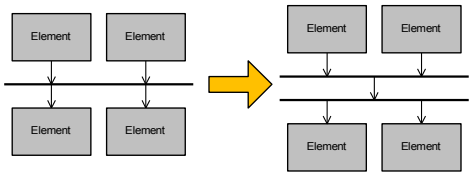
If a time event has not a predecessor and it has has more than one successor, then the time event is replaced by an initial node with a fore node as predecessor and this fork node will get the predecessors of the time event.



Preparation Rule adPR$_{2_12}$	
If a time event has a predecessor and has not a successor, then it is replaced by a flow final.	

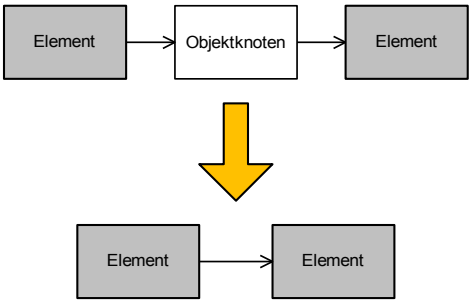
Preparation Rule adPR$_{2_13}$	
If a time event has more than one predecessor and has not a successor, then it is replaced by a merge node with a flow final as successor.	

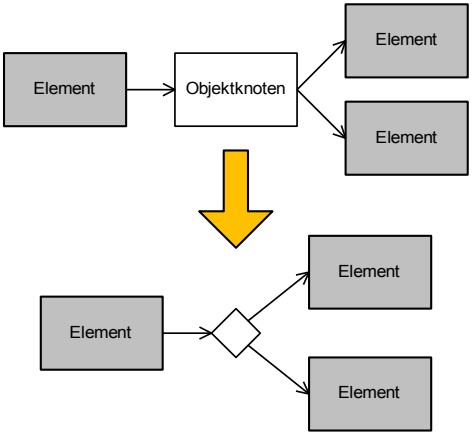
Preparation Rule adPR$_{2_14}$	
If a merge node has more than one predecessor and more than one successor, than the merge node will get a decision node as successor and every successor of the merge node will become successors of the decision node.	

Preparation Rule adPR$_{2_15}$	
If a join node has more than one predecessor and more than one successor, than the join node will get a fork node as successor and every successor of the join node will become successors of the fork node.	

Preparation Rule adPR$_{2_16}$	
If an object node or a data store has not a predecessor and it has a successor, that it is removed.	

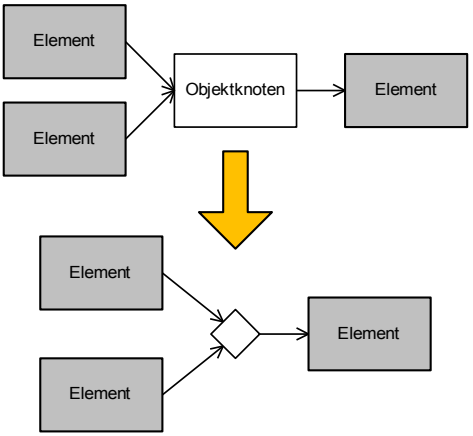
Preparation Rule adPR$_{2_17}$	
If an object node or a data store has a predecessor and it has not a successor, that it is removed.	

Preparation Rule adPR$_{2_18}$	
If an object node or a data store has a predecessor and it has a successor, that it is removed.	

Preparation Rule adPR$_{2_19}$	
If an object node or a data store has a predecessor and it has more than one successor, then it is replaced by a decision node.	

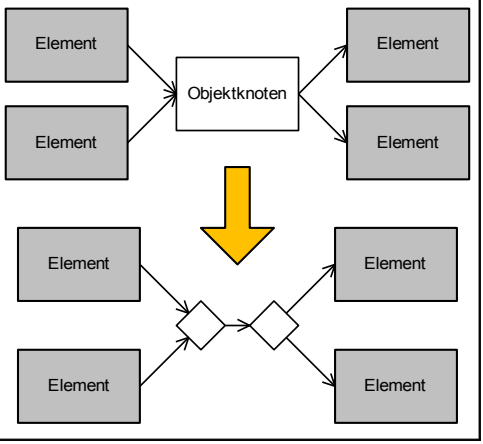
Preparation Rule adPR$_{2_20}$

If an object node or a data store has a predecessor and it has more than one successor, then it is replaced by a merge node.



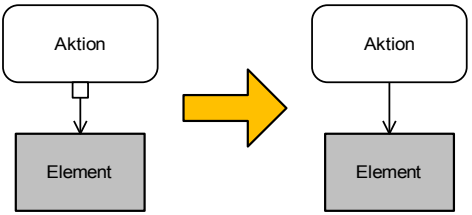
Preparation Rule adPR$_{2_21}$

If an object node or a data store has more than one predecessor and it has more than one successor, then it is replaced by a merge node and a decision node that will get the successors of the element.



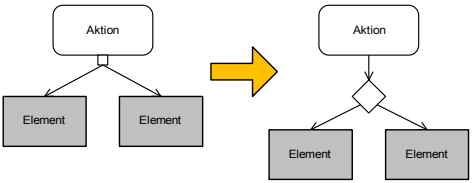
Preparation Rule adPR$_{2_22}$

If an object node pin has a successor, then it will be removed.



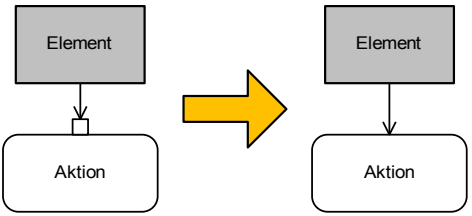
Preparation Rule adPR$_{2_23}$

If an object node pin has more than one successor, then the action will get a decision node as successor and all successors of the object node pin will become successors of the decision node.



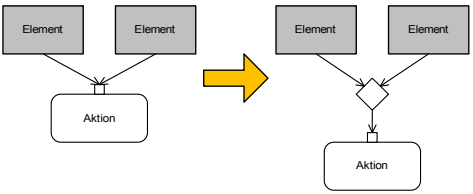
Preparation Rule adPR$_{2_24}$

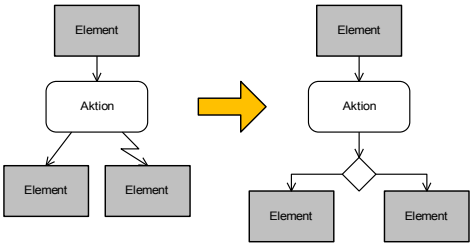
If an object node pin has a predecessor, then it will be removed.

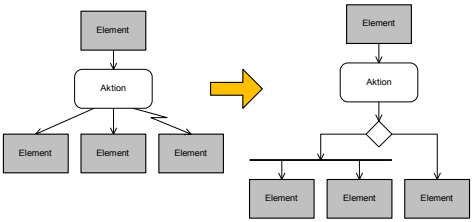


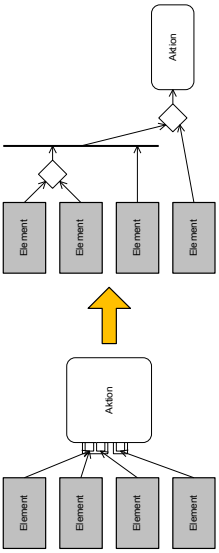
Preparation Rule adPR$_{2_25}$

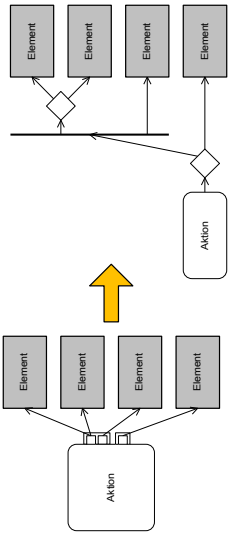
If an object node pin has more than one predecessor, the action will get a merge node as predecessor and all predecessors of the object node pin will become predecessors of the merge node.



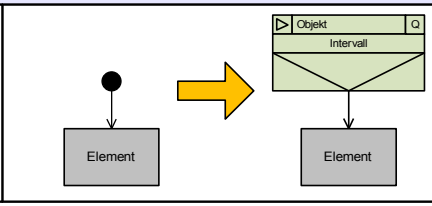
Preparation Rule adPR$_{2_26}$	
If an action has a successor and an exception handler, then the action gets a decision node as successor and the successor and exception handler will become successors of the decision node.	

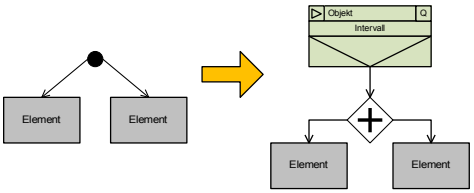
Preparation Rule adPR$_{2_27}$	
If an action has more than one successor and an exception handler, then the action gets a decision node as successor and a new fore node, that will get the successors of the action as its successors and exception handler will become successors of the decision node.	

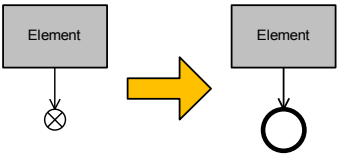
Preparation Rule adPR$_{2_28}$	
If an action has more than one parameter set with incoming process flows, then the action will get a merge node as its predecessor and if a parameter set has one object node pin, then it will become the predecessor of the merge node and if a parameter set has more than one object node pin, then they will get a join node as successor and the join node will become a predecessor of the merge node and if more than one element has an object node pin as their successors than these elements will get an additional merge node has their common successor.	

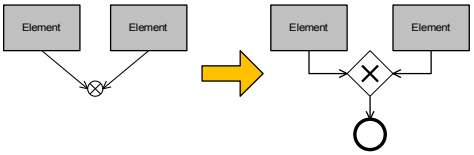
Preparation Rule adPR$_{2_29}$	
If an action has more than one parameter set with outgoing process flows, then the action will get a decision node as successor and if a parameter set has one object node pin, then it will become the successor of the decision node and if a parameter set has more than one object node pin, then they will get a fork node as predecessor and the fork node will become a successor of the decision node and if an object node pin has more then one successor than these elements will get an additional decision node has their common predecessor.	

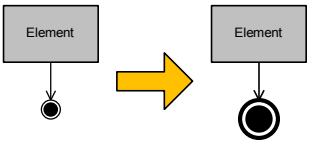
A.3.2 UML AD Transformationsregeln

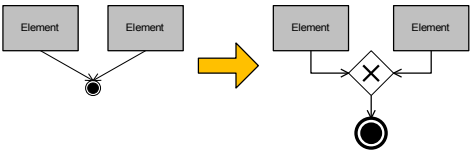
Transformation Rule $adTR_1$	
If an initial node has a successor, than it will be a source.	

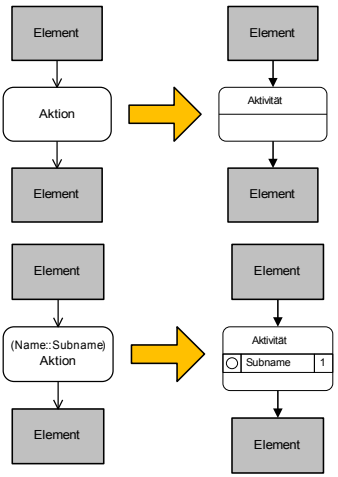
Transformation Rule $adTR_2$	
If an initial node has more than one successor, then it will be a source with a parallel gateway as its successor.	

Transformation Rule $adTR_3$	
If a flow final has a predecessor, than it will be a sink.	

Transformation Rule $adTR_4$	
If a flow final has more than one predecessor, than it will be a sink with an exclusive Gateway as predecessor.	

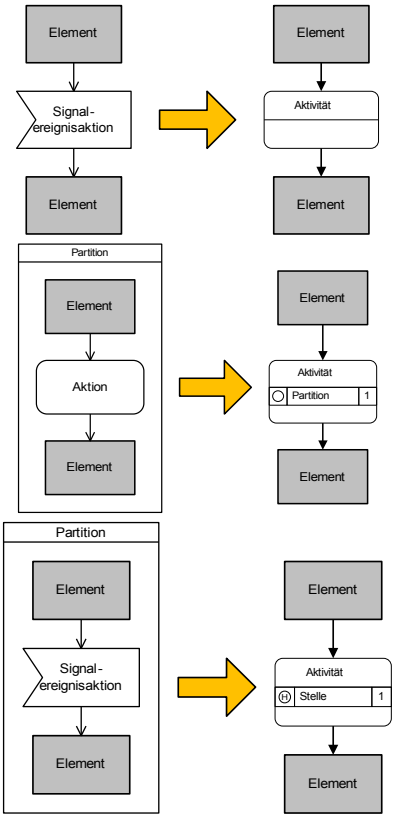
Transformation Rule adTR$_5$	
If an activity final has a predecessor, then it will be a terminating sink.	

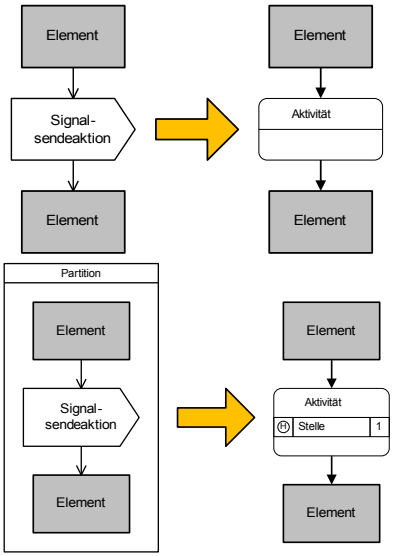
Transformation Rule adTR$_6$	
If an activity final has more than one predecessor, then it will be a terminating sink with an exclusive Gateway as predecessor.	

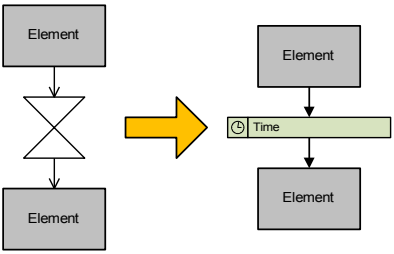
Transformation Rule adTR$_7$	
If the element is an action, then it will be an activity and if the action is placed in a partition, then the deepest partition is added as a resource or if the action contains the information about a partition, then the deepest partition is added as a resource.	

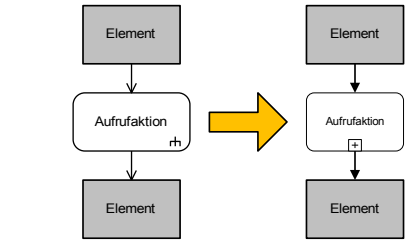
Transformation Rule adTR$_8$

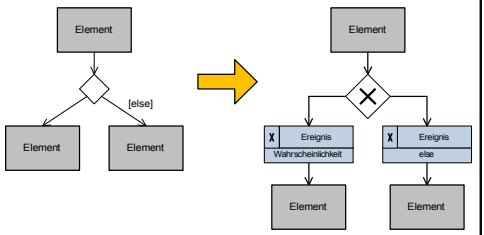
If the element is a signal event action, then it will be an activity and if the signal event action is placed in a partition, then the deepest partition is added as a resource.

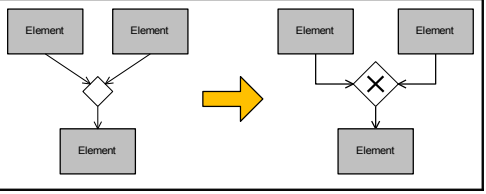


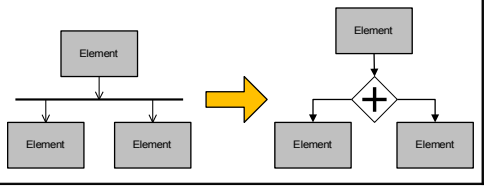
Transformation Rule adTR$_9$	
If the element is a signal send action, then it will be an activity and if the signal send action is placed in a partition, then the deepest partition is added as a resource.	

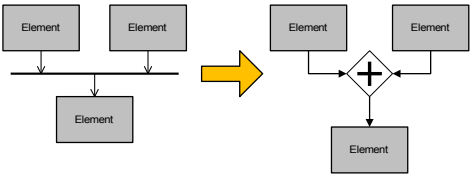
Transformation Rule adTR$_{10}$	
If the element is a time event, then it will be a delay.	

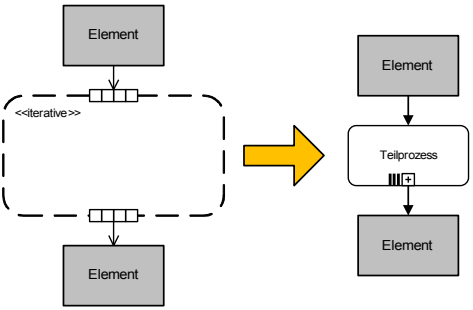
Transformation Rule adTR$_{11}$	
If the element is a call action, then it will be a sub process.	

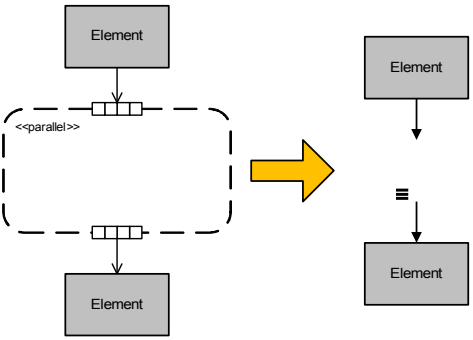
Transformation Rule adTR$_{12}$	
If the element is a decision node, then it will be an exclusive Gateway and every successor will get an exclusive key as successor that will be the successor of the exclusive Gateway and if a control flow is marked with an else, then the corresponding exclusive key will be marked with an else.	

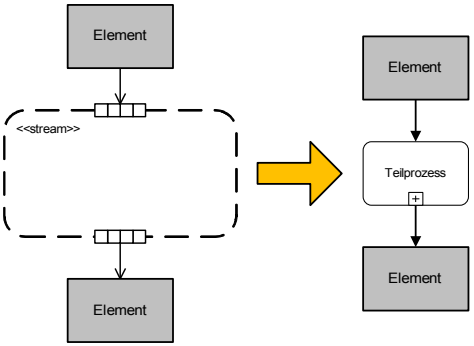
Transformation Rule adTR$_{13}$	
If the element is a merge node, then it will be an exclusive Gateway.	

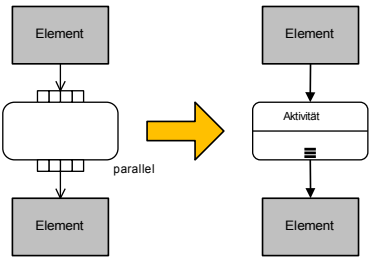
Transformation Rule adTR$_{14}$	
If the element is a fork node, then it will be a parallel Gateway.	

Transformation Rule adTR$_{15}$	
If the element is a join node, then it will be a parallel Gateway.	

Transformation Rule adTR$_{16}$	
If an expansion region is marked as iterative, then it will be a sub process marked with a sequential marker.	

Transformation Rule adTR$_{17}$	
If an expansion region is marked as parallel, then it will be a sub process marked with a parallel marker.	

Transformation Rule adTR$_{18}$	
If an expansion region is marked as stream, then it will be a sub process.	

Transformation Rule adTR$_{19}$	
If an expansion region is modeled in the shorthand notation, than it will be an activity with a parallel marker and if the expansion region is placed in a partition, then the deepest partition is added as a resource.	

Anhang B: Klassendefinitionen für AnyLogic

Klasse: DecisionMap

```
import java.util.Hashtable;
import java.util.Random;

public class DecisionMap implements java.io.Serializable {
  private static final long serialVersionUID = 1L;

  private Hashtable<String, DecisionTable> decisionTables;
  private static DecisionMap decisionMap;

  private DecisionMap() {
    this.decisionTables = new Hashtable<String, DecisionTable>();
  }

  public boolean addDecisionTable(String id, DecisionTable table) {
    if ( table.valid() ) {
      this.decisionTables.put(id, table);
      return true;
    } else {
       return false;
    }
  }

  public DecisionTable getDecicionTable(String id) {
     if ( this.decisionTables.containsKey(id) == false )
        return null;

    return decisionTables.get(id);
  }

  public static DecisionMap getInstance() {
    if ( decisionMap == null ) {
      decisionMap = new DecisionMap();
    }
    return decisionMap;
  }

  public String getPath(String id) {
     if ( this.decisionTables.containsKey(id) == false ) {
        return null;
     }

     DecisionTable table = decisionTables.get(id);

     float probability = new Random().nextFloat() % 1;
     return table.getPaths(probability);
  }
}
```

Klasse: DecisionTable

```
public class DecisionTable implements java.io.Serializable {
  private static final long serialVersionUID = 1L;
  private Vector<Case> cases;

  public DecisionTable() {
     this.cases = new Vector<Case>();
  }

  public Vector<Case> getCases() {
   return this.cases;
  }

  public void addCase(Case newCase) {
     cases.add(newCase);
  }

  public boolean valid() {
     float probability = 0.0f;

     for ( int i=0; i<cases.size(); i++ ) {
       probability += cases.get(i).getProbability();
     }

     if ( probability == 1.0f )
       return true;
    else
       return false;
  }

  public String getPaths(float probability) {
     if ( probability < 0.0f && probability > 1.0f )
      return null;

    int index = 0;

    for ( float p = 0.0f; p<probability; index++ ) {
      p += this.cases.get(index).getProbability();
    }

    return this.cases.get(index-1).getPath();
  }
}
```

Klasse: ProSiTEntity

```
import java.util.Hashtable;

public class ProSiTEntity extends
         com.xj.anylogic.libraries.enterprise.Entity
         implements java.io.Serializable {

  private static final long serialVersionUID = 1L;
  private Hashtable<String, String>  pathTable;
  private ProSiTEntity original;

  public ProSiTEntity() {
     pathTable     = new Hashtable<String, String>();
     positionTable = new Hashtable<String, Integer>();
  }

  public void copy(ProSiTEntity entity) {
    this.pathTable = entity.pathTable;
     this.original = entity;
}

  public void setOrDecision(String decision, String paths) {
    pathTable.put(decision, paths);
  }

  public boolean hasPath(String decision, char path) {
    if ( pathTable.containsKey(decision) == false ) {
      return false;
    }

    String paths = pathTable.get(decision);

    for ( int i=0; i<paths.length(); i++ ) {
      char nextPath = paths.charAt(i);
        if ( nextPath == path ) {
          return true;
        }
    }

    return false;
  }
}
```

Klasse: Case

```
public class Case implements java.io.Serializable {
  private static final long serialVersionUID = 1L;
  private String path;
  private float  probability;

  public Case(String path, float probability) {
    this.path = path;
    this.probability = probability;
  }

  public String getPath() {
     return path;
  }

  public void setPath(String path) {
    this.path = path;
  }

  public float getProbability() {
     return probability;
  }

  public void setProbability(float probability) {
    this.probability = probability;
  }

}
```

Klasse: EverytimeTable

```
public class EverytimeTable implements java.io.Serializable {
  private static final long serialVersionUID = 1L;
  private Vector<Long> entryIds;


  public EverytimeTable(){
    this.entryIds = new Vector<Long>();
  }

  public void addId(long entryId) {
    if ( hasId(entryId) == true )
      return;

    entryIds.addElement(new Long(entryId));
  }

  public boolean hasId(long entryId) {
    for ( int i=0; i<entryIds.size(); i++ ) {
      if ( entryIds.get(i).longValue() == entryId )
        return true;
      }
    return false;
  }
}
```

Klasse: EverytimeMap

```
public class EverytimeMap implements java.io.Serializable {
  private static final long serialVersionUID = 1L;

  private Hashtable<String, EverytimeTable> everytimeTables;
  private static EverytimeMap everytimeMap;

  private EverytimeMap() {
    this.everytimeTables = new Hashtable<String, EverytimeTable>();
  }

  public static EverytimeMap getInstance() {
    if ( everytimeMap == null ) {
      everytimeMap = new EverytimeMap();
    }
    return evertimeMap;
  }

  public boolean checkEntityId(String decision, long entryId) {
    EverytimeTable table = everytimeTables.get(decision);
    if ( table == null ) {
      table = new EverytimeTable();
      table.addId(entryId);
      everytimeTables.put(decision, table);

      return false;
    }

    if ( table.hasId(entryId) == false ) {
      table.addId(entryId);
      return false;
    } else
      return true;
  }
}
```

Literaturverzeichnis

Aguilar, Marc; Rautert, Tankred; Pater, Alexander J.G (1999) Business Process Simulation : A fundamental step supporting process centred management. In: Farrington, P.A.; Nembhard, H.B.; Sturrock, D.T.; Evans, G.W. (Hrsg.) Winter Simulation Conference 1999. Phoenix, S. 1383-1392.

Allweyer, Thomas (2005) Geschäftsprozessmanagement - Strategie, Entwurf, Implementierung, Controlling. W3L, Herdecke.

Allweyer, Thomas (2009) BPMN 2.0 Business Process Model and Notation – Einführung in den Standard für die Geschäftsprozess-modellierung. 2. Auflage, Books on Demand GmbH, Norderstedt.

Almeder, Christian; Preusser, Margaretha (2007) A toolbox for simulation-based optimization of supply chains. In: Henderson, Shane G.; Biller, Bahar; Hsieh, Ming-Hua; Shortle, John; Tew, Jeffrey D.; Barton, Russel R. (Hrsg.) Winter Simulation Conference 2007. Washington, S. 1932-1939.

An, Lianjun; Jeng, Jun-Jang (2005) On developing system dynamics model for business process simulation. In: Kuhl, Michael E.; Steiger, Natalie M.; Armstrong, F. Brad; Joines, Jeffrey A. (Hrsg.) Winter Simulation Conference 2005. Orlando, S. 2068-2077.

Ardagna, Danilo; Cappiello, Cinzia; Lovera, Marco; Pernici, Barbara; Tanelli, Mara (2008) Active Energy-Aware Management of Business-Process Based Applications : Position Paper. In: Mähönen, Petri; Pohl, Klaus; Priol, Thierry (Hrsg.) First European Conference, ServiceWave2008, Madrid. S. 183-195.

Arlow, Jim; Neustadt, Ila (2005) UML 2 and the Unifed Process : Practical Object Oriented Analysis and Design. 2. Auflage, Pearson Education, Upper Saddle River.

Bangsow, Steffen (2008) Fertigungssimulation mit Plant Simulation - Anwendungen und Programmierung mit Beispielen und Lösungen. Carl Hanser Verlag, München.

Banks, Jerry; Carson II, John S.; Nelson, Barry L.; Nicol, David M. (2005) Discrete Event System Simulation. 4. Aufl. Pearson Prentice Hall, Upper Saddle River.

Barber, K.D.; Dewhurst, F.W.; Burns, R.L.D.H.; Rogers, J.B.B. (2003) Business-process modelling and simulation for manufacturing management : A practical way forward. In: Business Process Management Journal 9 (4), S: 527-542.

Barton, Russel R.; Fishwick, Paul A.; Sargent, Robert G.; Henrikson, James O.; Twomey, Janet M. (2003) Simulation - Past, Present and Future. Chick, S.; Sánchez, P. J.; Ferrin, D.; Morrice, D. J. (Hrsg.) Winter Simulation Conference 2003. New Orleans, S. 2044-2050.

Bassenge, Gero (2002) Automatische Klassifizierung von Wortformen in Texten der deutschen Gegenwartssprache. Dissertation, Technische Universität Darmstadt.

Becker, Jörg; Mathas, Christoph; Winkelmann, Axel (2009) Geschäftsprozessmanagement. Springer, Berlin.

Becker, Jörg; Schütte, Reinhard (2004) Handelsinformationssysteme – Domänenorientierte Einführung in die Wirtschaftsinformatik. 2. Aufl., Redline Wirtschaft, Frankfurt.

Davenport, Thomas H. (1993) Process innovation : reengineering work through information technology. Harvard Business School Press, Bosten.

Decker, Gero; Tscheschner, Willi; Puchan, Jörg (2009) Migration von EPK zu BPMN. Nüttgens, Markus; Rump, Frank J.; Mendling, Jan; Gehrke, Nick (Hrsg.) EPK 2009 - Geschäftsprozessmanagement mit Ereignisgesteuerten Prozessketten. Berlin, S. 91-109.

Delfmann, Patrick; Herwig, Sebastian; Lis, Łukasz; Stein, Armin (2008) Eine Methode zur formalen Spezifikation und Umsetzung von Bezeichnungskonventionen für fachkonzeptionelle Informationsmodelle. In: Loos, Peter; Nüttgens, Markus; Turowiski, Klaus; Werth, Dirk (Hrsg.) Modellierung betrieblicher Informationssysteme (MobIS 2008) – Modellierung zwischen SOA und Compliance Management. Saarbrücken, S. 23-38.

Delfmann, Patrick; Herwig, Sebastian; Lis, Lukasz (2009) Konfliktäre Bezeichnungen in Ereignisgesteuerten Prozessketten : Linguistische Analyse und Vorschlag eines Lösungsansatzes. In: Nüttgens, Markus; Rump, Frank J.; Mendling, Jan; Gehrke, Nick (Hrsg.) EPK 2009 - Geschäftsprozessmanagement mit Ereignisgesteuerten Prozessketten. Berlin, S. 178-194.

Desel, Jörg; Erwin, Thomas (2000) Modeling, Simulation and Analysis of Business Processes. In: van der Aalst, Wil .M.P; Desel, Jörg; Oberweis, Andreas (Hrsg.) Business Process Management : Model, Techniques, and Empirical Studies. Springer, Berlin, S. 129-141.

Dickmann, Christoph; Klein, Harald; Birkhölzer, Thomas; Fietz, Wolfgang; Vaupel, Jürgen; Meyer, Ludger (2007) Deriving a Valid Process Simulation from Real World Experiences. In: Wang, Qing; Pfahl, Dietmar; Raffo, David M. (Hrsg.) International Conference on Software Process, ICSP 2007. Minneapolis, S. 272-282

DIN (2005) Qualitätsmanagementsysteme - Grundlagen und Begriffe. DIN EN ISO 9000:2005, Deutsches Institut für Normung, Berlin.

Ehm, Hans; McGinnis, Leon; Rose, Oliver (2009) Are Simulation Standards in out Future? In: Dunkin, Ann; Ingalls, Ricki G.; Yücesan, Enver; Rossetti, Manuel D.; Hill, Ray; Johansson, Björn (Hrsg.): Winter Simulation Conference 2009. Austin, S. 1695-1702.

Fetter, R.B.; Thompson, J. D. (1965) The Simulation of Hospital Systems. In: Operations Research 41 (10), S. 689-711.

Fettke, Peter (2006) State-of-the-Art des State-of-the-Art : Eine Untersuchung der Forschungsmethode „Review" innerhalb der Wirtschaftsinformatik. In Wirtschaftsinformatik 48 (4), S. 257-266.

Forrester, Jay W. (1972) Grundzüge einer Systemtheorie. Gabler, Wiesbaden.

Frank, Ulrich; van Laak, Bodo L. (2003) Anforderungen an Sprachen zur Modellierung von Geschäftsprozessen. Arbeitsberichte des Instituts für Wirtschaftsinformatik, Universität Koblenz Landau, http://www.wi-inf.uni-duisburg-essen.de/FGFrank/documents/Arbeitsberichte_Koblenz/Nr34.pdf.

Freund, Jakob; Rücker, Bernd; Henninger, Thomas (2010) Praxishandbuch BPMN. Hanser, München.

Friedrich, Dieter (1984) Systemtheorie und ökonomische Modelle: Einführung in systemtheoretische Grundlagen, Konzeptionen und Methoden der Wirtschaftstheorie und Ökonometrie. Haufe, Freiburg.

Friedrich, Fabian (2009) Measuring Semantic Label Quality Using WordNet. In: Nüttgens, Markus; Rump, Frank J.; Mendling; Jan; Gehrke, Nick (Hrsg.) EPK 2009 - Geschäftsprozessmanagement mit Ereignisgesteuerten Prozessketten. Berlin, S. 7-21.

Funk, Burkhardt; Gómez; Jorge Marx; Niemeyer, Peter; Teuteberg, Frank (2010) Geshäftsprozessintegration mit SAP : Fallstudien zur Steuerung von Wertschöpfungsprozessen entlang der Supply Chain. Springer, Berlin.

Gadatsch, Andreas (2010) Grundkurs Geschäftsprozess-Management. 6. Aufl. Vieweg, Wiesbaden.

Glinz, Martin (2008) Modellierung in der Lehre an Hochschulen : Thesen und Erfahrungen. In: Informatik Spektrum 31 (5), S. 425-434.

Gräning, André; Felden, Carsten; Piechocki, Maciej (2011) Status Quo und Potenziale der eXtensible Business Reporting Language für die Wirtschaftsinformatik. In Wirtschaftsinformatik 53 (4), S. 225-234.

Greasley, Andrew (2000) Effective Uses Of Business Process Simulation. In: Joines, J. A.; Barton, Russel R.; Kang, K.; Fishwick, Paul A. (Hrsg.) Winter Simulation Conference 2000. Orlando, S. 2004-2009.

Greasley, Andrew (2003) Using business-process simulation within a business-process reengineering approach. In: Business Process Management Journal 9 (4), S. 408-420.

Greiffenberg, Steffen (2003) Methoden als Theorien der Wirtschaftsinformatik. In: Uhr, Wolfgang; Esswein, Werner; Schoop, Eric (Hrsg.) Internationale Tagung Wirtschaftsinformatik. Dresden, S. 947–968.

Gruhn, Volker; Pieper, Daniel; Röttgers, Carsten (2006) MDA : Effektives Software-Engineering mit UML 2 und Eclipse. Springer, Berlin.

Heavey, Cathal; Ryan, John (2006) Process modelling support for the conceptual modelling phase of a simulation project. In: Perrone, L. Felipe; Lawson, Barry G.; Liu, Jason; Wieland, Frederick P. (Hrsg.) Winter Simulation Conference 2006. Monterey, S. 801-808.

Heinrich, Lutz J.; Heinzl, Armin; Roithmayr, Friedrich (2007) Wirtschaftsinformatik : Einführung und Grundlegung. 3. Auflage, Oldenbourg, München.

Helbig, Gerhard; Buscha, Joachim (2004) Leitfaden der deutschen Grammatik. 5. Aufl., Langenscheidt, Berlin.

Heß, Michael; Meis, Jochen (2011) Entwurf ausgewählter Spracherweiterungen zur Ressourcenmodellierung in Pflegedienstleistungsmodellen. In: Bernstein, Abraham; Schwabe, Gerhard (Hrsg.) Proceedings of the 10th International Conference on Wirtschaftsinformatik – Volume 1, S. 99-108.

Himmelreich, Katja (2007) Modellierung und Simulation der Prozesse in einer Notfallaufnahme eines Krankenhauses. Diplomarbeit an der Technischen Universität Ilmenau, Betreuer: Hagen Schorcht.

Houy, Constantin; Fettke, Peter; Loos, Peter (2009) Stilisierte Fakten der Ereignisgesteuerten Prozesskette - Anwendung einer Methode zur Theoriebildung in der Wirtschaftsinformatik. In: Nüttgens, Markus; Rump, Frank J.; Mendling, Jan; Gehrke, Nick (Hrsg.) EPK 2009 - Geschäftsprozessmanagement mit Ereignisgesteuerten Prozessketten. Berlin, S. 42-57.

Hoyer, Volker; Bucherer, Eva; Schnabel, Florian (2007) Collaborative e-Business Process Modelling - Transforming Private EPC to Public BPMN Business Process Models. In: Hofstede, Arthur ter; Benatallah, Boualem; Paik, Hye-Young (Hrsg.) Business Process Management Workshops - BPM 2007 International Workshops BPI, BPD, CBP, ProHealth, RefMod, semantics4ws. Brisbane, S. 185-196.

Hlupic, Vlatka (2000) Simulation Software - An operational research society survey of academic and industrial users. In: Joines, J. A.; Barton, Russel R.; Kang, K.; Fishwick, Paul A. (Hrsg.) Winter Simulation Conference 2000. Orlando ,S. 1676–1683.

Jansen-Vullers, Monique; Netjes, Mariska (2006) Business Process Simulation – A Tool Survey. In: Jensen, Kurt (Hrsg.) Seventh Workshop and Tutorial on Practical Use of Coloured Petri Nets and the CPN Tools. Aarhus, S. 77-96.

Jähnke, Norman (2009) Prozessanalyse Im Medizinischen Umfeld - Ist-Analyse und Entwicklung von Optimierungsansätzen am Beispiel einer Augenambulanz. Diplomarbeit an der Technischen Universität Ilmenau, Betreuer: Hagen Schocht.

Jansen-Vullers, Monique; IJpelaar, Rob; Loosschilder, Maurice (2006) Workflow Patterns modelled in Arena. Technische Universität Eindhoven.

Just, Maximilian (2010) Simulation von Geschäftsprozessen mit einer Simulationsumgebung für Produktionssysteme. Bachelorarbeit an der Technischen Universität Ilmenau, Betreuer: Oliver Kloos.

Känel, von Siegfried (1972) Einführung in die Kybernetik für Ökonomen. 2. Auflage, Die Wirtschaft, Berlin.

Kecher, Christoph (2009) UML 2 : das umfassende Handbuch. 4. Auflage, Galileo Press, Berlin.

Keller, Gerhard; Nüttgens, Markus; Scheer, August-Wilhelm (1992) Semantische Prozessmodellierung auf der Grundlage "Ereignisgesteuerter Prozessketten (EPK)". http://www.iwi.uni-sb.de/Download/iwihefte/heft89.pdf, Abruf am 2008-08-27 (Veröffentlichungen des Instituts für Wirtschaftsinformatik (IWi), Universität des Saarlandes Heft 89).

Kondratyev, Mikhail; Garifullin, Maxim (2009) Parallel Discrete Event Simulation with AnyLogic. In: Malyshkin, Victor (Hrsg.) 10th International Conference, PaCT 2009. Novosibirsk, S. 226-236.

Kelton, W. David; Sadowski, Randall P.; Sturrock, David T. (2007) Simulation with Arena. 4. Auflage, McGraw-Hill, Boston.

Kloos, Oliver; Nissen Volker; Petsch, Mathias (2009) From Process to Simulation - A Transformation Model Approach. In: Esswein, Werner; Mendling, Jan; Rinderle-Ma, Stefanie (Hrsg.) 3rd International Workshop on Enterprise Modelling and Information Systems Architectures. Ulm, S. 83-96.

Kloos, Oliver (2010) Geschäftsprozessmanagement Übungsheft. Technische Universität Ilmenau.

Kloos, Oliver; Nissen, Volker (2010) Vom Prozess zur Simulation - ein Transformationsmodell-Ansatz. In: Claus, Thorsten; Herrmann, Frank (Hrsg.) Workshop "Simulation als betriebliche Entscheidungshilfe" im Rahmen der 14. ASIM Fachtagung "Simulation in Produktion und Logistik", Karlsruhe, S. 105-119.

Kloos, Oliver; Schorcht Hagen; Petsch, Mathias; Nissen, Volker (2010) Dienstleistungsmodellierung als Grundlage für eine Simulation. In: Thomas, Oliver; Nüttgens, Markus (Hrsg.) Dienstleistungsmodellierung – Interdisziplinäre Konzepte und Anwendungsszenarien. Berlin, Springer, S. 91-111.

Kloos, Oliver; Nissen, Volker; Petsch, Mathias; Schorcht, Hagen (2011) Service Modelling as a Basis for Simulation. In: Enterprise Modelling and Information Systems Architectures – An International Journal 6 (2), S. 21-34.

Krieger, David J. (1996) Einführung in die allgemeine Systemtheorie. Fink, München.

Kuhn, Thomas S. (1993) Die Struktur wissenschaftlicher Revolution. Suhrkamp, Frankfurt.

Lam, Kokin; Lau, R.S.M (2004) A simulation approach to restructuring call centers. In: Business Process Management Journal 10 (4), S. 481-494.

Landau, Ariel; Wasserkrug, Segev; Gilat, Dagan; Razinkov, Natalia; Sela, Aviad; Aiber, Sarel (2004) A Methodological Framework for business-Oriented Modeling of IT Infrastructure. In: Ingalls, Ricki G.; Rossetti, Manuel D.; Smith, J.S.; Peters, B.A. (Hrsg.) Winter Simulation Conference 2004. Washington, S. 474-484.

Law, Averill M. (2004) Simulation Modelling and Analysis. 4. Auflage, McCraw-Hill, Bosten.

Lehner, Franz (1999) Theoriebildung in der Wirtschaftsinformatik. In: Becker, Jörg; König, Wolfgang; Schütte, Reinhard; Wendt, Oliver; Zelewski, Stephan (Hrsg.) Wirtschaftsinformatik und Wissenschaftstheorie: Bestandsaufnahme und Perspektiven. Gabler, Wiesbaden, S. 5-24.

Lehner, Franz; Hildebrand, Knut; Maier, Ronald (1995) Wirtschaftsinformatik: Theoretische Grundlagen. Hanser, Wien.

Leopold, Henrik; Smirnov, Sergey; Mendling, Jan (2009) On Labeling Quality in Business Process Models. In: Nüttgens, Markus; Rump, Frank J.; Mendling; Jan; Gehrke, Nick (Hrsg.) EPK 2009 - Geschäftsprozessmanagement mit Ereignisgesteuerten Prozessketten. Berlin, S. 42-57.

Leyking, Katrina; Angeli, Ralf (2008) Model-based Competency-oriented Business Process Analysis. In: Loos, Peter; Nüttgens, Markus; Turowiski, Klaus; Werth, Dirk (Hrsg.) Modellierung betrieblicher Informationssysteme (MobIS 2008) – Modellierung zwischen SOA und Compliance Management. Saarbrücken, S. 39-57.

Li, bin; Li, Wen-feng (2010) Modeling and Simulation of container Terminal Logistics Systems Using Harvard Architecture and Agend-Based Computing. In: Johansson, Björn; Jain, Sanjay; Montoya-Torres, Jairo; Hugan, Joe; Yücesan, Enver (Hrsg.) Winter Simulation Conference 2010. Baltimore, S. 3396-3410.

Lloret, Jaime; Garcia-Sabater, Jose P.; Marin-Garcia, Juan A. (2009) Cooperative Supply Chain Re-scheduling : The Case of an Engine Supply Chain. In: Luo, Yuhua (Hrsg.) 6th International Conference, CDVE 2009. Luxemburg, S. 376-383.

Loos, Peter; Fettke, Peter; Weißenberger, Barbara E.; Zelewski, Stephan; Heinzl, Armin; Frank, Ulrich; Iivari, Juhani (2011) Welche Rolle spielen eigentlich stilisierte Fakten in der Grundlagenforschung der Wirtschaftsinformatik? In: Wirtschaftsinformatik 53 (2), S. 109-121

Lowery, Julie C. (1998) Getting Started in Simulation in Healthcare. In: Medeiros, D.J; Watson, E.F; Carson, J.S; Manivannan, M.S (Hrsg.) Winter Simulation Conference 1998. Washingston D.C., S. 31-35.

Marmor, Yariv N.; Wasserkrug, Segev; Zeltyn, Sergey; Mesika, Yossi; Greenshpan, Ohad; Carmeli, Boaz; Shtub, Avraham; Mandelbaum, Avishai (2009) Toward Simulation-Based Real-Time Decision-Support Systems for Emergency Departments. In: Dunkin, Ann; Ingalls, Ricki G.; Yücesan, Enver; Rossetti, Manuel D.; Hill, Ray; Johansson, Björn (Hrsg.) Winter Simulation Conference 2009. Austin, S. 2042-2053.

Mendling, Jan; Nüttgens, Markus (2003a) EPC Syntax Validation with XML Schema Languages. In: Nüttgens, Markus; Rump, Frank J. (Hrsg.) EPK 2003 Geschäftsprozessmanagement mit Ereignis-gesteuerten Prozessketten. Bamberg, S. 19-30.

Mendling, Jan; Nüttgens, Markus (2003b) EPC Modeling based on Implicit Arc Types. In: Godlevsky, Mikhail; Liddle, Stephen W.; Mayr, Heinrich C. (Hrsg.) International Conference ISTA 2003, Kharkiv, S. 131-142.

Mendling, Jan; Nüttgens, Markus (2006) EPC markup language (EPML): An XML-based interchange format for event-driven process chains (EPC) In: Information Systems and E-Business Management 4 (3), S. 245-263.

Mendling, Jan (2008) Metrics for Process Models : Empirical Foundations of Verification, Error Prediction, and Guidelines for Correctness. Springer, Berlin.

Mendling, Jan; Reijers, Hajo A. (2008) The Impact of Activity Labeling Styles on Process Model Quality. In: Hesse, Wolfgang; Oberweis, Andreas(Hrsg.) Proceedings of the Third AIS SIGSAND European Symposium on Analysis, Design, Use and Societal Impact of Information Systems, Marburg S. 117-127.

Mendling, Jan; Reijers, Hajo A.; Recher, Jan C. (2010) Activity labeling in process modeling : empirical insights and recommendations. In: Information Systems 35 (4). S. 467-482.

Meyer, Ruth; Page, Bernd; Kreutzer, Wolfgang; Knaak, Nicolas; Lechler, Tim (2005) DESMO-J – A Framework for Discrete Event Modelling & Simulation. In: Page, Bernd; Kreutzer, Wolfgang (Hrsg.) The Java Simulation Handbook - Simulating Discrete Event Systems with UML and Java. Shaker, Aachen, S. 263-335

Miller, George A. (1995) WordNet : A Lexical Database for English. In: Communications of the ACM 38 (11), S. 39-41.

Neumann, Stefan; Rosemann, Michael; Schwegmann, Ansgar (2005) Simulation von Geschäftsprozessen. In: Becker, Jörg; Kugeler, Martin; Rosemann, Michael (Hrsg.) Prozessmanagement - Ein Leitfaden zur prozessorientierten Organisationsgestaltung. 5. Aufl., Springer, Berlin, S. 435-453.

Nitsch, Peter (2010) Simulation zur Verbesserung von Geschäftsprozessen. Diplomarbeit an der Technischen Universität Ilmenau, Betreuer: Hagen Schocht.

Object Management Group (2009) Business Process Modelling Notation Version 1.1. http://www.omg.org/spec/BPMN/1.1/PDF, Abruf am 2009-07-20.

Object Management Group (2010) OMG Unified Modeling Language (OMG UML) Superstructure - Version 2.3. http://www.omg.org/spec/UML/2.3/Superstructure/PDF, Abruf am 2011-09-12.

Object Management Group (2011) Business Process Model and Notation (BPMN) Version 2.0. http://www.omg.org/spec/BPMN/2.0/PDF, Abruf am 2011-04-28.

Oestereich, Bernd; Weiss, Christian; Schröder, Claudia; Weilkiens, Tim; Lenhard, Alexander (2004) Objektorientierte Geschäftsprozessmodellierung mit der UML. Dpunkt, Heidelberg.

Oestereich, Bernd (2006) Analyse und Design mit UML 2.1 : Objektorientierte Softwareentwicklung. 8. Auflage, Oldenburg, München.

Österle, Hubert; Becker, Jörg; Frank, Ulrich; Hess, Thomas; Karagiannis, Dimitris; Krcmar, Helmut; Loos, Peter; Mertens, Peter; Oberweis, Andreas; Sinz, Elmar J. (2010a) Memorandum der gestaltungsorientierten Wirtschaftsinformatik. In: Zeitschrift für betriebswirtschaftliche Forschung 62 (6), S. 664-672.

Österle, Hubert; Winter, Robert; Brenner, Walter (2010b) Gestaltungsorientierte Wirtschaftsinformatik : Ein Plädoyer für Rigor und Relevanz. Book-on-demand.

Overhage, Sven; Schlauderer, Sebastian; Borkmeier, Dominik (2011) Sind Ereignisgesteuerte Prozessketten besser für Fachanwender geeignet als UML-Aktivitätsdiagramme? - Eine empirische Untersuchung. In: Bernstein, Abraham; Schwabe, Gerhard (Hrsg.) Proceedings of the 10th International Conference on Wirtschaftsinformatik – Volume 2. Zürich, S. 745–755.

Page, Bernd (1991) Diskrete Simulation: Eine Einführung mit Modula-2. Springer, Berlin.

Page, Bernd; Kreuzer, Wolfgang (2005) The Java Simulation Handbook – Simulating Discrete Event Systems with UML and Java. Shaker, Aachen.

Paul, Ray J.; Hlupic, Vlatka; Giaglis, George M. (1998) Simulation Modelling of Business Processes. In: Avison, D.; Edgar-Neville, D. (Hrsg.) 3rd U.K. Academy of Information Systems Conference. Lincoln, S. 311-320.

Petsch, Mathias; Schorcht, Hagen; Nissen, Volker; Himmelreich, Katja (2008) Ein Transformationsmodell zur Überführung von Prozessmodellen in eine Simulationsumgebung. In: Loos, Peter; Nüttgens, Markus; Turowiski, Klaus; Werth, Dirk (Hrsg.) Modellierung betrieblicher Informationssysteme (MobIS 2008) – Modellierung zwischen SOA und Compliance Management. Saarbrücken, S. 209-219.

Popper, Karl L. (1995) Objektive Erkenntnis: ein evolutionärer Entwurf. Hoffmann und Campe, Hamburg.

Profozich, David M.; Sturrock, David T. (2005) Introduction to Siman/Cinema. In: Kuhl, Michael E.; Steiger, Natalie M.; Armstrong, F. Brad; Joines, Jeffrey A. (Hrsg.) Winter Simulation Conference 2005. Orlando, S. 515-518.

Riege, Christian; Saat, Jan; Bucher, Tobias (2009) Systematisierung von Evaluationsmethoden in der gestaltungsorientierten Wirtschaftsinformatik. In: Becker, Jörg; Krcmar, Helmut; Niehaves, Björn (Hrsg.) Wissenschaftstheorie und gestaltungsorientierte Wirtschaftsinformatik. Physica, Heidelberg, S. 69-86.

Rittgen, Peter (1999) Modified EPCs and their semantics. In: Frank, Ulrich; Hampe, Felix (Hrsg.) Arbeitsberichte des Instituts für Wirtschaftsinformatik, Nr. 19. Universität Koblenz-Landau, Koblenz.

Rosemann, Michael (1996) Komplexitätsmanagement in Prozeßmodellen: Methodenspezifische Gestaltungsempfehlungen für die Informationsmodellierung. Gabler, Wiesbaden.

Rossetti, Manuel D. (2010) Simulation Modeling and Arena. Hoboken, Wiley, 2010.

Ryan, John; Heavey, Cathal (2006) Process modeling for simulation. In: Computers in Industry 57 (5), S. 437-450.

Sharawi, Abeer; Sala-Diakanda, Serge N.; Dalton, Adam; Quijada, Sergio; Yousef, Nabeel; Rabelo, Luis; Sepulveda, Jose (2006) A Distributed Simulation Approach for Modeling and Analyzing Systems of Systems. In: Perrone, L. Felipe; Lawson, Barry G.; Liu, Jason; Wieland, Frederick P. (Hrsg.) Winter Simulation Conference 2006. Monterey, S. 1028-1035.

Sarshar, Kamyar; Loos, Peter (2005) Comparing the Control-Flow of EPC and Petri Net from the End-User Perspective. In: van der Aalst, Wil; Bentallah, Boualem; Casati, Fabio; Curbera, Francisco (Hrsg.) Business Process Management 3rd International Conference, Nancy, S. 434-439.

Scheer, August-Wilhelm (1991) Architektur integrierter Informationssysteme. Springer, Berlin.

Scheer, August-Wilhelm (1997) Wirtschaftsinformatik - Referenzmodelle für industrielle Geschäftsprozesse. Springer, Berlin.

Schmid, Helmut (1994) Probabilistic Part-of-Speech Tagging Using Decision Trees. In: International Conference on New Methods in Language Processing, Manchester. http://www.ims.uni-stuttgart.de/ftp/pub/corpora/net-tagger.ps.gz, Abruf: 2011-08-29.

Schmid, Helmut (1995) Improvements In Part-of-Speech Tagging With an Application to German. In: EACL SIGDAT Workshop, Dublin. http://www.ims.uni-stuttgart.de/ftp/pub/corpora/tree-tagger2.pdf, Abruf: 2011-08-29

Schmietendorf, Andreas; End, Andreas (2009) A Prototypical Simulation Model to Analysze the Business Process Performance. In: Abran, Alain; Braungarten, René; Dumke, Reiner R.; Cuadrado-Gallego, Juan J.; Brunekreef, Jacob (Hrsg.) International conferences IWSM 2009 and Mensura 2009, Amsterdam, S. 130-143.

Schmidt, Johannes (2010) Bezeichnungen in eEPK-Modellen. Proseminararbeit an der Technischen Universität Ilmenau, Betreuer: Oliver Kloos.

Schönherr, Oliver; Rose, Oliver (2009) First Steps Towards a General SysML Model for Discrete Processes in Production Systems. In: Dunkin, Ann; Ingalls, Ricki G.; Yücesan, Enver; Rossetti, Manuel D.; Hill, Ray; Johansson, Björn (Hrsg.) Winter Simulation Conference 2009. Austin,S. 1711-1718.

Schönherr, Oliver; Rose, Oliver (2010) Modellierung mit SysML zur Abbildung von Produktionsprozessen. In: Zülich, Gert; Stock, Patricia (Hrsg.) Integrationsaspekte der Simulation : Technik, Organisation und Personal. 14. ASIM Fachtagung, Karlsruhe, S. 453-460.

Schorcht, Hagen; Petsch, Mathias; Nissen, Volker; Himmelreich, Katja (2007) Entwicklung eines konzeptuellen Modells für die Überführung von Prozessmodellen in Simulationsmodelle. In: Koschke, Rainer; Herzog, Ottheim; Rödiger, Karl-Heinz; Ronthaler, Marc (Hrsg.) Informatik 2007, Bremen. S. 413-415.

Schütte, Reinhard; Siedentopf, Jukka; Zelewski, Stephan (1999) Wirtschaftsinformatik und Wissenschaftstheorie: Grundpositionen und Theoriekerne. Universität GH Essen, Institut für Produktion und Industrielles Informationsmanagement, Essen.

Seyfert, Wolfgang; Kavermann, Ansgar (2006) Mit Simulation die Geschäftsprozesse einer Notaufnahme gestalten. In: Biethahn, Jörg (Hrsg.) Simulation als betriebliche Entscheidungshilfe: Neuere Werkzeuge und Anwendungen aus der Praxis. Georg-August-Universität, Institut für Wirtschaftsinformatik, Göttingen, S. 31-49.

Shannon, Robert E. (1992) Introduction to Simulation. In: Swain, James J.; Goldsman, David; Crain, Robert C.; Wilson, James R. (Hrsg.) Winter Simulation Conference 1992. Arlington, S. 65-73.

Stachowiak, Herbert (1974) Allgemeine Modelltheorie. Springer, Berlin.

Staud, Josef L. (2005) Datenmodellierung und Datenbankentwurf : Ein Vergleich aktueller Methoden. Springer, Berlin.

Staud, Josef L. (2006) Geschäftsprozessanalyse: Ereignisgesteuerte Prozessketten und objektorientierte Geschäftsprozessmodellierung für Betriebswirtschaftliche Standardsoftware. 3. Aufl. Springer, Berlin.

Staud, Josef L. (2010) Unternehmensmodellierung : Objektorientierte Theorie und Praxis mit UML 2.0. Springer, Berlin.

Störrle, Harald (2006) A Comparison of (e)EPCs and UML 2 Activity Diagrams. In: Nüttgens, Markus; Rump, Frank J.; Mendling, Jan (Hrsg.) EPK 2006 - Geschäftsprozessmanagement mit Ereignis-gesteuerten Prozessketten. Gesellschaft für Informatik, Bonn, S. 177-188.

Tumay, Kerim (1996) Business Process Simulation. In: Charnes, John M.; Morrice, Douglas J.; Brunner, Daniel T.; Swain, James J. (Hrsg.) Winter Simulation Conference 1996. Coronado, S. 93-98.

Vanderhaeghen, Dominik; Zang, Sven; Hofer, Anja; Adam, Otmar (2005) XML-based Transformation of Business Process Models – Enabler for Collaborative Business Process Management. In: Nüttgens, Markus; Mendling, Jan (Hrsg.) Proceedings of the Workshop XML4BPM 2005, Karlsruhe, S. 81-94.

van der Aalst, Wil .M.P; ter Hofstede, Arthur H.M; Kiepuszewski, Bartek; Barros, A. P. (2003) Workflow Patterns. In: Distributed and Parallel Databases 14, S. 5–51.

van der Aalst, Wil; Nakatumba, J.; Rozinat, A.; Russel, N. (2010) Business Process Simulation: How to get it right? In: Brocke, Jan vom; Rosemann, Michael (Hrsg.) International Handbook on Business Process Management 1 : Introduction, Methods, and Information Systems. Springer, S. 313-337.

VDI (1995a) Simulation von Logistik-, Materialfluß- und Produktions-systeme - Begriffsdefinitionen. VDI-Richtlinie 3633, Verein Deutscher Ingenieure, Berlin.

VDI (1995b) Simulation von Logistik-, Materialfluß- und Produktionssysteme - Grundlagen. VDI-Richtlinie 3633, Verein Deutscher Ingenieure, Berlin.

Vetter, Max (1994) Informationssysteme in der Unternehmung: eine Einführung in die Datenmodellierung und Anwendungsentwicklung. Teubner, Stuttgart.

Wilson, Brian (1990) Systems: concepts, methodologies and applications. 2. Aufl. John Wiley & Sons, Chichester.

Wilde, Thomas; Hess, Thomas (2006) Methodenspektrum der Wirtschaftsinformatik : Überblick und Portfoliobildung. Arbeitsbericht Nr. 2/2006, Ludwig-Maximilians-Universität München, Institut für Wirtschaftsinformatik und Neue Medien. http://www.wim.bwl.uni-muenchen.de/download/epub/ab_2006_02.pdf, Abruf am: 2011-09-06

Witte, Thomas (1973) Simulationstheorie und ihre Anwendungen auf betriebliche Systeme. Gabler, Wiesbaden.

Zelewski, Stephan (2009) Wirtschaftsinformatik und Wissenschaftstheorie : Zwischen Konformität und organisiertem Wildwuchs. In: Becker, Jörg; Krcmar, Helmut; Niehaves, Björn (Hrsg.) Wissenschaftstheorie und gestaltungsorientierte Wirtschaftsinformatik. Physica, Heidelberg S. 225-243.